Humphries

ISSN 0265-2919
90p 56

THE HOME COMPUTER ADVANCED COURSE

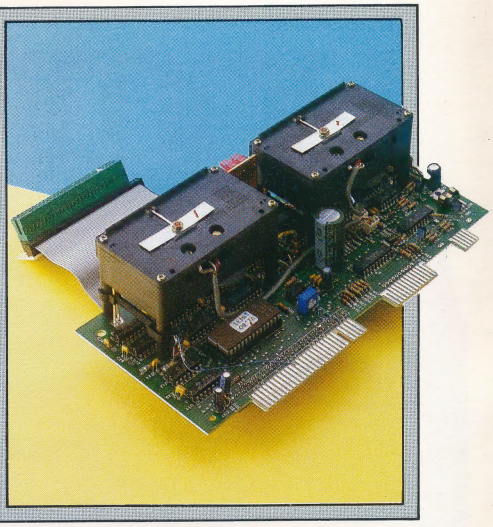
MAKING THE MOST OF YOUR MICRO

An ORBIS Publication
IR£1.15 Aus $2.15 NZ $2.65 SA R2.45 Sing $4.50

CONTENTS

Next Week

- **The Rotronics Wafadrive for the Spectrum provides an alternative mass storage system to Sinclair's own Microdrives. We look at how the Wafadrive measures up against its rival**
- **We begin a new series that takes an in-depth look at the Commodore 64's operating system**
- **Young children often amaze their parents with the ease in which they learn to use computers. Our education series looks at child programmers**



QUIZ

1) Which Prestel service is aimed especially at education?
2) Is a daughter board vital to the running of a microcomputer?
3) The 'hero' of Knight Lore has been featured in two other Ultimate games. Name one of them.
4) What is the function of the Intel 8087 chip?

Answers To Last Week's Quiz

1) Because it is the only one that is 'programmable', the Memocon Crawler is the one Movit that can be considered a true 'robot'.
2) A 'database manager' is the program that holds the information entered into it. This information is known as the 'database'.
3) The event addresses are stored on the stack, where they can be easily accessed by the A register.
4) PASCAL's three iteration constructs are: WHILE...DO, REPEAT...UNTIL and FOR...DO.

Editor Stephen Cooke; **Art Editor** Claudia Zeff; **Production Editor** Bobby Pickering; **Technical Editor** Steve Colwill; **Designer** Julian Dorr; **Art Assistant** Liz Dixon; **Staff Writer** Stephen Malone; **Sub Editor** Jonathan Kaye; **Contributors** Geoff Bains, Nick Walsh, Joe Pritchard, Steve Malone, David Mudd, Anthony Ginn, Steve Colwill; **Software Consultants** Pilot Software City; **Group Art Director** Perry Neville; **Managing Director** Stephen England; **Published by Orbis Publishing Ltd: Editorial Director** Brian Innes; **Project Development** Peter Brookesmith; **Executive Editor** Maurice Geller; **Production Assistant** Alastair Gourlay; **Subscription Manager** Christine Allen; **Designed and produced by Bunch Partworks Ltd; Editorial Office** 14 Rathbone Place, London W1P 1DE; **© APSIF Copenhagen 1985; © Orbis Publishing Ltd 1985: Typeset by Universe; Reproduction by Mullis Morgan Ltd; Printed in Great Britain by Heanor Gate Printing Ltd., Derby**

HOW TO OBTAIN ISSUES AND BINDERS FOR THE HOME COMPUTER ADVANCED COURSE – Issues can be obtained by placing an order with your newsagent or direct from our subscription department. If you have any difficulty obtaining any back issues from your newsagent, please write to us stating the issue(s) required and enclosing a cheque for the cover price of the issue(s). **AUSTRALIA** – please write to: Gordon & Gotch (Aus) Ltd, 114 William Street, PO Box 767G, Melbourne, Victoria 3001. **MALTA, NEW ZEALAND & SOUTH AFRICA –** Back numbers are available at cover price from your newsagent. In case of difficulty, write to the address given for binders. **UK/EIRE** – Price: 90p/IR£1.15. Subscription: 6 months: £26.00. 1 Year: £52.00. Binder: please send £3.95 per binder, or take advantage of our special offer in early issues. **EUROPE** – Price: 90p. Subscription: 6 months air: £44.72. Surface: £36.14. 1 year air: £89.44. Surface: £72.28. Binder: £5.00. Airmail: £8.25. **MALTA** – Obtain binders from your newsagent or Miller (Malta) Ltd, MA Vassalli Street, Valetta, Malta. Price: £3.95. **MIDDLE EAST** – Price: 90p. Subscription: 6 months air: £50.18. Surface: £36.14. 1 year air: £100.36. Surface: £72.28. Binder: £5.00. Airmail: £8.25. **AMERICAS/ASIA/AFRICA** – Price: US/CAN$1.95/90p. Subscription: 6 months air: £59.54. Surface: £36.14. 1 year air: £119.08. Surface: £72.28. Binder: £5.00. Airmail: £9.50. **SOUTH AFRICA** – Price: SA R2.45. Obtain binders from any branch of Central News Agency or Intermag, PO Box 57394, Springfield 2137. **SINGAPORE** – Price: Sing $4.50. Obtain binders from MPH Distributors, 601 Sims Drive, 03-07-21, Singapore 1438. **AUSTRALASIA/FAR EAST** – Price: 90p. Subscription: 6 months air: £64.22. Surface: £36.14. 1 year air £128.44. Surface: £72.28. Binder: £5.00. Airmail: £9.75. **AUSTRALIA** – Price: AusS2.15. Obtain binders from First Post Pty Ltd, 23 Chandos Street, St Leonards, NSW 2065. **NEW ZEALAND** – Price: NZ$2.65. Obtain binders from your newsagent or Gordon & Gotch (NZ) Ltd, PO Box 1595, Wellington.
ADDRESS FOR BINDERS AND BACK ISSUES – Orbis Publishing Limited, Orbis House, Bedfordbury, London WC2 4BT. Telephone 01-379 5211. Cheques/postal orders should be made payable to Orbis Publishing Limited. Binder prices include postage and packing and prices are in stirling. Back issues are sold at the cover price, and we do not charge carriage in the UK.
NOTE – Binders and back issues are obtainable subject to availability of stocks. Whilst every attempt is made to keep the price of the issues and binders constant, the publishers reserve the right to increase the stated prices at any time when circumstances dictate. Binders depicted in this publication are those produced for the UK and Australian markets only. Binders and Issues may be subject to import duty and/or local taxes, which are not included in the above prices unless stated.
ADDRESS FOR SUBSCRIPTIONS – Orbis Publishing Limited, Hurst Farm, Baydon Road, Lambourn Woodlands, Newbury Berks, RG16 7TW. Telephone: 0488-72666. All cheques/postal orders should be made payable to Orbis Publishing Limited. Postage and packaging is included in subscription rates, and prices are given in sterling.

COVER PHOTOGRAPHY BY MARCUS WILSON — SMITH

FACTS ON FILE

One of the greatest advantages of using computers in education is that they provide access to large and speedily revised databases. We look at the beneficial aspects of incorporating databases into the classroom and examine a specifically designed program for the young pupil.

There are many educational benefits in obtaining information from an 'electronic library': access is immediate, it is not necessary to leave the room, or even one's seat, to retrieve data. This eliminates the possibility of someone 'having the book out'. Furthermore, computerised data can easily be updated, whereas the information held in a textbook is static and can only be changed by reprinting. The current costs of book production represent a serious problem in the provision of source material and schools are frequently forced to supply out-of-date publications. More significantly, the high cost of the printed word can also impede the spread of new educational methods. In a class equipped with computers, changing the curriculum requires changing only the master disks — completely restocking the library is made redundant.

The educational benefits are made particularly apparent inhe combination of microcomputer and telecommunications for on-line database services. To have, say, the Encyclopaedia Britannica on disk would be an extremely useful asset. If the information you wanted could be found by searching the ordinary indexes, in alphabetical order, then taking a copy of the book off the shelf would be perfectly adequate. But if you wanted to find all the *burrowing* marsupials, for example, or list all those countries where the percentage of the

The World About Them
Despite fears that the use of computers in schools would confine children to the narrow world of the VDU, practice has shown that database software, in particular, can lead to all sorts of 'off-keyboard' activities. The need to gather information for entry into the database can be tackled in many different ways, including that of the 'nature walk', where students are encouraged to observe, record and classify elements of the environment

STEVE CROSS

GNP spent on education is above a certain level — well, to compile it by hand from the text could be very tiresome.

Encyclopaedia Britannica, however, runs to 26 volumes and the storage space and comprehensive search facilities necessary would require an extemely large amount of processing power; which is where on-line services come into their own. You can tap into them with a standard 300- or 1200-baud modem and telephone line for access to a staggering array of information. The Inspec database, for example, compiled by the Institute of Electrical Engineers, covers the whole range of electronics and electrical engineering, with over two million entries. The Royal Society of Chemists produces Chemical Engineering Abstracts listing research papers, articles and other reference material. While the printed versions are worth having for keeping up with research, it's easier to search for specific references on-line.

Dialog, in the US, is the world's largest on-line database service and one of the most widely used in the educational field. It contains 200 bases featuring over 100 publications and 100 million items of information — abstracts on everything from art history to zoology. As you search Dialog, you will be referred not only to the author and title of any relevant papers or articles, but you will also be given an abstract offering quite a comprehensive description and, often enough, containing all the information you will require. You can then, of course, print this out, or else have the entire printed piece sent to you, usually at a fairly reasonable cost. The number of on-line databases available is being matched by a growing amount of database-searching software to help users extract and sort information.

A different sort of approach is offered by Prestel with its 'Schools Link'. This offers schools information on all aspects of computing and creates a forum for ideas and contacts. It is aimed at teachers, rather than pupils, and contains reviews of software, educational robots, books and suggestions for projects; it also facilitates educational programs to be downloaded into the teacher's, or school's, machine.

However, in addition to the 'electronic library', databases are becoming steadily more popular in the classroom, and playing an active role. Several programs already exist that give school children the experience of handling information and structuring it according to their own individual needs, allowing children the opportunity to gather, input and manipulate their own data.

THE FACTFILE DATABASE

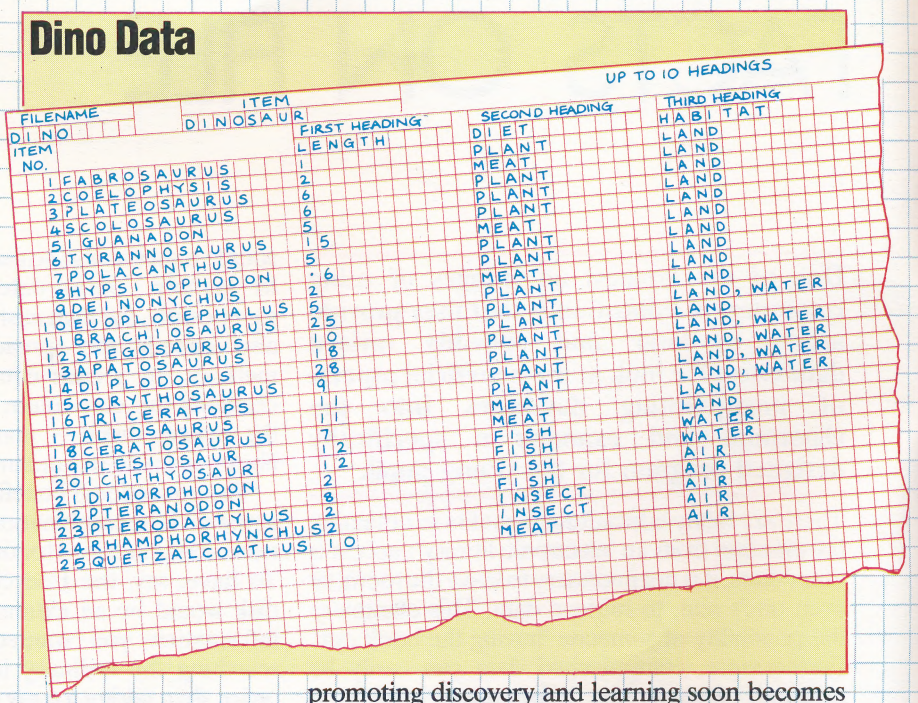
The possibilities of using databases were discussed at an educational conference at Cambridge University in 1981, and the debate inspired the creation of a program called Factfile, which enables very young children to build data files on any topic they choose.

The active role of a database like Factfile in promoting discovery and learning soon becomes apparent. Let us suppose that the children want to create a database holding information about their teachers. First, they will have to think about the categories in which they are going to organise the information. This will depend on what they already know and what they can discover for themselves. They may, for example, decide to include age, height, subject, sex, and 'type of teacher'. Having determined the structure of the database they must then gather information for the specified categories. At this point, the exercise develops into a diverse educational operation — at the very least, the children will have to gather physical information. If measurements are needed they will have to determine the units to be used and the data to be included. The process becomes more complex as the children begin to understand the meaning of the categories they themselves have generated. The field 'Type of teacher', requires a good deal of discussion. Does it mean: 'Which subject do they teach?' or 'Is the teacher nice or nasty?'

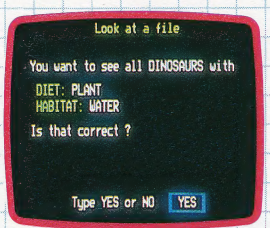
When they have finished compiling the database, other children may question it, correlating information from more than one category. Perhaps they will want to list all the female teachers who have been classified in the category 'Nice' or all the male mathematics teachers. The learning process is then taken a step further as the children begin to analyse the results. Factfile allows children to save their database onto disk or cassette and add more information at a later date, enabling them to return to the files whenever it may be necessary.

One criticism of children using computers has been that the medium encourages them to sit in front of a VDU pursuing activities with no relevance to the world in which they live. But besides giving children direct experience of databases, Factfile leads into a myriad off-

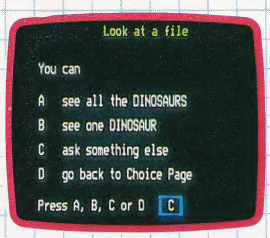
Dino Data



The Memory Remains
Factfile is a complete database system for younger students that provides an excellent introduction to database management. Supplied with the program is a sample file, called Dino, which contains information on dinosaurs. Children are encouraged to familiarise themselves with Dino and to catalogue the contents of the file (as above) before moving on to the creation of their own databases

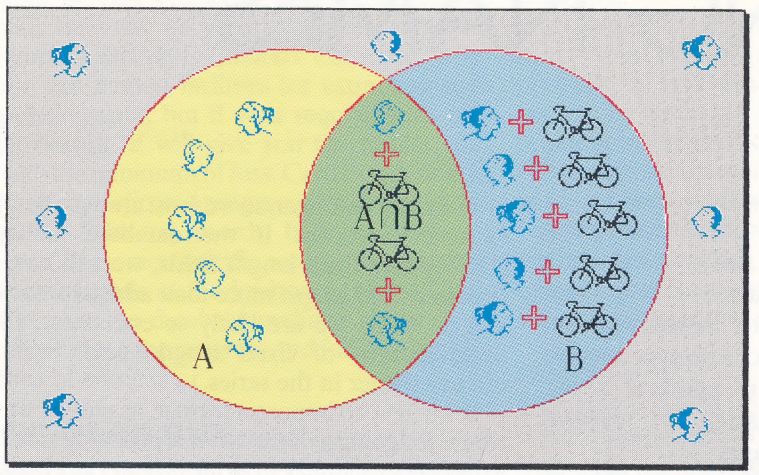


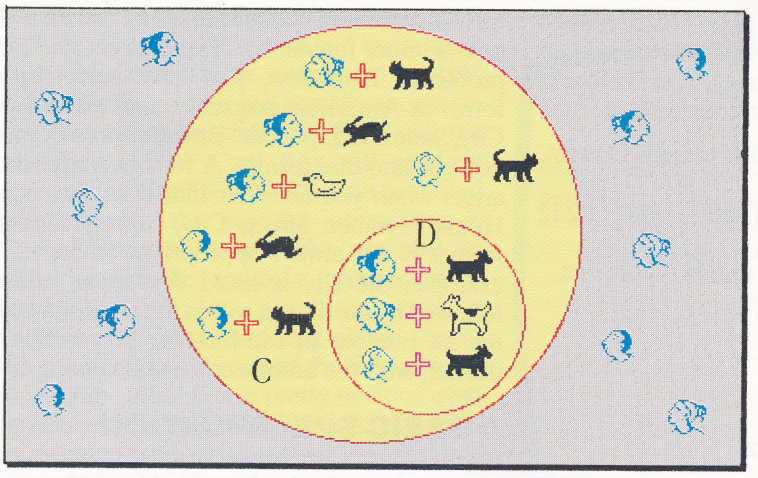
Search Request



Dino Menu

Set Work





IAN McKINNELL

keyboard activities. If a file was created with details of local fauna, the children would have to carefully record the details of each plant and then develop their own classification system in order to build the database. One school in London constructed a database as part of their environmental studies program, using the results of a survey with local people as input.

Another popular package, designed for younger children, is Your Facts. The children are asked their names, if they are girl or boy, if they have a pet, a watch, a bicycle and either a brother or a sister. When the information for one child has been keyed in, the program asks if anyone else would like to type in their information. There is room for facts on 40 children, so the whole class can be included. When all the information has been keyed in, and 'No' has been typed in reply to 'WOULD ANYONE ELSE LIKE TO MAKE AN INPUT?', the children are returned to the menu, where they may see all the records, find which children fall in a particular category — such as those who have a watch, or those who have a bicycle — or they can play a game. The game consists of the computer figuring out a child's name after asking various questions 'ARE YOU A BOY?', 'DO YOU HAVE A WATCH?', and so on. It then asks, for example, 'IS YOUR NAME TIMOTHY?'. Your Facts is an ideal introduction, for young children, to the concept of a database. It encourages reading and writing, relates to them personally and is an enjoyable exercise.

Classified Information

The construction and interrogation of databases is not unrelated to other work done by children in schools. Modern mathematics syllabuses introduce the idea of sets at an early age. Set theory is largely concerned with the classification of data and the connections between such classifications. Setting up a database is in many ways directly analogous to this, categorising properties of an entry into fields and records so that the database can be interrogated to yield relationships between items.

The concepts of intersection and subset in particular have real meaning in database interrogation, corresponding to the creation of short-lists. Intersection is the 'overlap' between two sets — for example, the intersection of the set of blond-haired children and the set of children with bicycles is the set of blond-haired children who own bicycles. A special case of intersection is where one set overlaps entirely with another, forming a subset, as in the second example

ONE WORLD PROGRAM

An Australian software house, Active Learning Systems, has produced an extensive database package of information on nearly every country in the world. The program is called One World and includes teachers' notes and worksheets for the individual and class. It contains over 30 pieces of information on each country, including facts about the political system, imports, exports, languages, neighbours, religion, literary rate, background and treaties. The percentage of rural and urban population is given, as well as percentages of the workforce in primary production, manufacturing and service industries. The proportions of desert, forest and cultivated land are also recorded. What makes the program so attractive, as with most databases, is its facilities to search for and analyse the data.

The main menu gives a series of options:

1. DISPLAY ANY COUNTRY
2. SEARCH USING UNIQUE INFORMATION
3. ANALYSE USING VARIOUS CRITERIA
4. HELP REFERENCE
5. EXIT

If a child knew of a Mr Shamir, a leader in the Middle East, and wanted to find his country, the second option would be selected. The SEARCH DATA menu would allow the child to type in his name under 3.LEADER OF GOVERNMENT and 4. HEAD OF STATE. The program would then inform the child that YITZHAK SHAMIR was the head of state of ISRAEL.

The ANALYSE option is the program's most powerful aspect. It allows children to analyse information, using up to three options, from 20 criteria. For example, they could find out in which European countries most of the population live in the countryside. This would only use two options: region and percentage of population. If the children heard that bauxite miners had gone on strike in Surinam, they could quickly discover the country's location, that bauxite is a major export and that the dispute could have a devastating effect on the economy.

One World teaches research and data interpretation skills, and encourages children to draw conclusions from their analyses of data. It is a great support to history, geography and social studies lessons.

A database can indicate to children most of what is available to learn, and encourages them to search for the answers. They can observe how the computer manipulates information, and understand the importance of asking the right questions and properly organising their input. Those who, after experiencing the use of a database, enter higher education and eventually the workforce, will surely possess an important and relevant skill.

NAMING NAMES

We have seen how databases can manipulate large bodies of related data given the fundamentals of proper structuring. In our second instalment on databases, we will construct a simple system based on the idea of an address book, highlighting the importance of careful planning in the design stage.

One of the major advantages of an old-fashioned address book is that very little structure is imposed beyond basic alphabetic ordering. It is flexible in the sense that you can have entries such as:

PETER GLOVER, 16 Rhiwbinal Cresent, Cardiff (0222-601227)
— his girlfriend's called Clair — office: 0222 680545 Ext 160
— call after 4 — moves to new flat in Jan.

Followed by:

GODFREY — 696-1949

Followed by:

GREG — see Ashton-Tate

Using an address book, you are free to list people under their first names, their surnames or even by 'indirect addressing' with 'pointers' to other entries. As soon as a database such as this gets transferred to a computer, however, a more systematic approach is usually called for.

Given that a computerised address book has some disadvantages compared with the original, let's design a format for one that will be flexible enough to cater for most circumstances. We'll consider each field in turn, starting with the name.

Names come in two parts: the generic half, called the surname, and the specific half, called the forename. In most countries, the forename(s) are followed by the surname, whereas in China, Japan, Hungary and some other countries, the surname precedes the forename. In Japan, there is only ever a single forename (except that it follows the surname), while in a Chinese community in England, it is customary to have an English forename as well as a Chinese forename. Thus, Li (surname) Yu Chow (forenames) will be known to most of his English friends as Paul.

Already, we have plenty of opportunity for confusion, and we're only considering the name field. Clearly, we will have to impose some discipline, and decide whether forenames or surnames are to come first, and how long an 'allowable' name may be. Names may be as short as Ng or as long as Cholmondley-Smythe, possibly even shorter or longer, so we must always allow for extreme examples. If the database manager (DBM) uses fixed length fields, we will have to choose a field length more than adequate for the longest name we are likely to encounter. (The disadvantages of fixed length fields will be discussed later in the series.)

If we have to impose a format, as indeed we do when designing a database, it is probably simplest to use the surname field as the primary key field, so we will start with the surname and follow it with one or more forenames. The address also poses problems. Doubtless, you have seen clip-out forms in American magazines that ask for your City, State and Zip Code. British addresses simply don't fit into that format. A further complication arises when you consider that even the 'Name, House Number, Street, City, County, Country' format is not always appropriate. In Japan, for example, the city is listed first in an address, followed by the district within the city, followed by the lot number of the building plot within the district, with the addressee's name coming last.

A BASIC SPECIFICATION

In this simple database example, as in more complex ones, it is probably not possible to cater for every conceivable combination, so we will have to settle for a compromise. Assuming you are never likely to be communicating with someone having 16 forenames, let's see how well the following basic skeleton would work:

- Surname Field — up to 40 characters
- Forename Field — up to 60 characters
- Address Field 1st Line — up to 80 characters
- Address Field 2nd Line — up to 80 characters
- Address Field 3rd Line — up to 80 characters
- Address Field 4th Line — up to 80 characters
- Address Field 5th Line — up to 80 characters
- Telephone Field — up to 20 characters
- Note Field — up to 80 characters

This basic specification should cover most eventualities, though there may still be problems. One potential difficulty is the limited length of the Note field. Another problem is that Country is not specified as a separate field. Depending on the length of the address, the country could be found in the third, fourth or fifth address fields, or not at all. This would not normally be a problem until we wanted to search our database using Country as a key. If your database application had to deal with many foreign correspondents, it might be better to design the database with a separate Country field. Decisions such as these will always be needed at the design stage.

The DBM you run on your computer will determine the ease with which you can implement a specific database on your system. One of the very simplest DBMs is Caxton's Card Box. The program has limited facilities for extracting and manipulating data, but if your requirements are straightforward, it will give you the results you need with the minimum of fuss. Card Box does not give you a sophisticated programming language to manipulate fields or records. It does, however, allow you to extract specific records by entering simple commands at the keyboard. Using Card Box to create the database outlined above, you need only to load the program and follow a simple sequence of entries.

USING CARD BOX

The first thing you have to do is decide on a format for the record. This format determines what information will be stored, how it will be indexed (that is, which fields will be designated as 'key' fields) and how the records will actually appear on the computer screen or printouts. If we call the database file ADBOOK, Card Box will create a format file called ADBOOK.FMT. This can be edited if required to alter the way the information is displayed. Alternative format files can also be created to display the database information in different ways.

As with most DBMs Card Box allows you to enter 'permanent' text in each of the fields. In the case of an address book, it is pretty obvious what the significance of each field is: John Smith is clearly a name and not part of an address; similarly, 0222-680545 is obviously a telephone number. In other databases, you might need to be reminded what the significance of each field is. This is where 'permanent' text comes in handy. Compare these two records:

06116
3995
86
34.75
Dongle with widget nozzle

and

MAKER'S PART NUMBER	06116
OUR PART NUMBER	3995
NUMBER LEFT IN STOCK	86
PRICE	34.75
DESCRIPTION	Dongle with widget nozzle

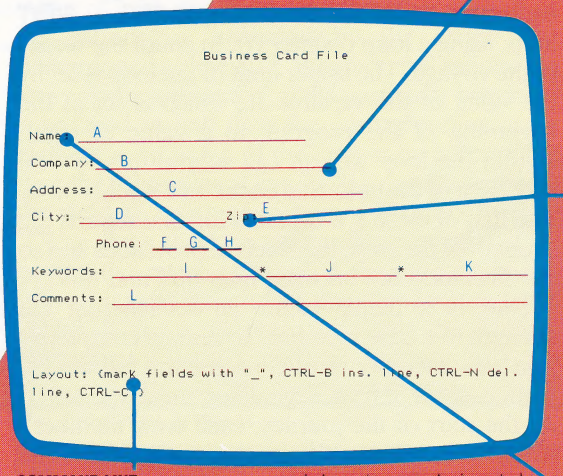
The information is identical in both cases, but the second example, with permanent text that appears in every record, makes mistakes much less likely.

Card Box allows each field to be given one of four possible indexing attributes: NONE, MAN(ual), AUTO or ALL. In a stock database, it is unlikely that we would ever need to use PRICE as a key field; you are not likely to ask: 'Do you have any parts that cost less than £40 and more than £30?' It is quite possible, however, that we would need to know how many #06116 items are still in stock, so that MAKER'S PART NUMBER would need to be made a key field.

Returning to our address book example, we are sure to want to search the database by Name, and probably by Forename too. Both of these would need to be made key fields. If we were running a business, we might also need to search records by City and possibly even by telephone area dialling code. It is important to remember that you cannot create an efficient database until you have worked out how you are going to use it. Unlike the old card indexes, databases require that you anticipate beforehand how you are likely to use them.

Design Elements

Megafinder allows you to design your own forms. This one was derived from the same form, supplied ready-made with the program, used in our other screen display

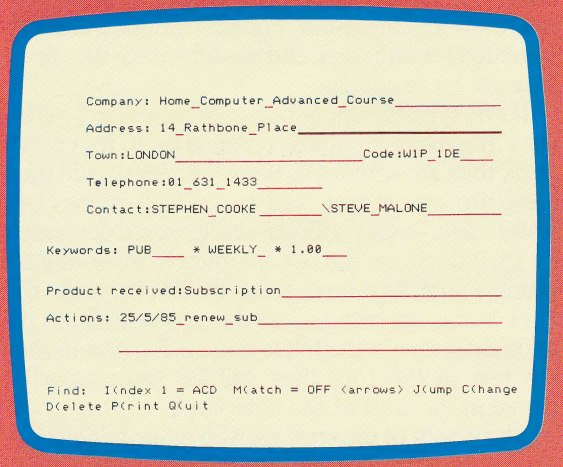


FIELD LENGTH
These are indicated by underline characters (_). The total length of all the fields in a form determines how many forms may be stored in a file. Therefore, it makes sense to keep fields as short as is practical, bearing in mind the nature of the data that will be entered

FIELD LABELS
These are entered by the user when designing the form and, in this program, are entirely for the user's convenience. The program uses them when displaying information on the screen but takes no account of field labels when sorting data, which it does using FIELD IDENTIFIERS

FIELD IDENTIFIERS
Not to be confused with the FIELD LABELS, the identifiers determine the order in which fields will be accessed when entering information. Identifiers also enable fields to be sorted separately or in groups. Using Megafinder's INDEX option, allocating field D to Index 1 would allow the user to sort quickly through the different records according to the alphabetical arrangement of cities

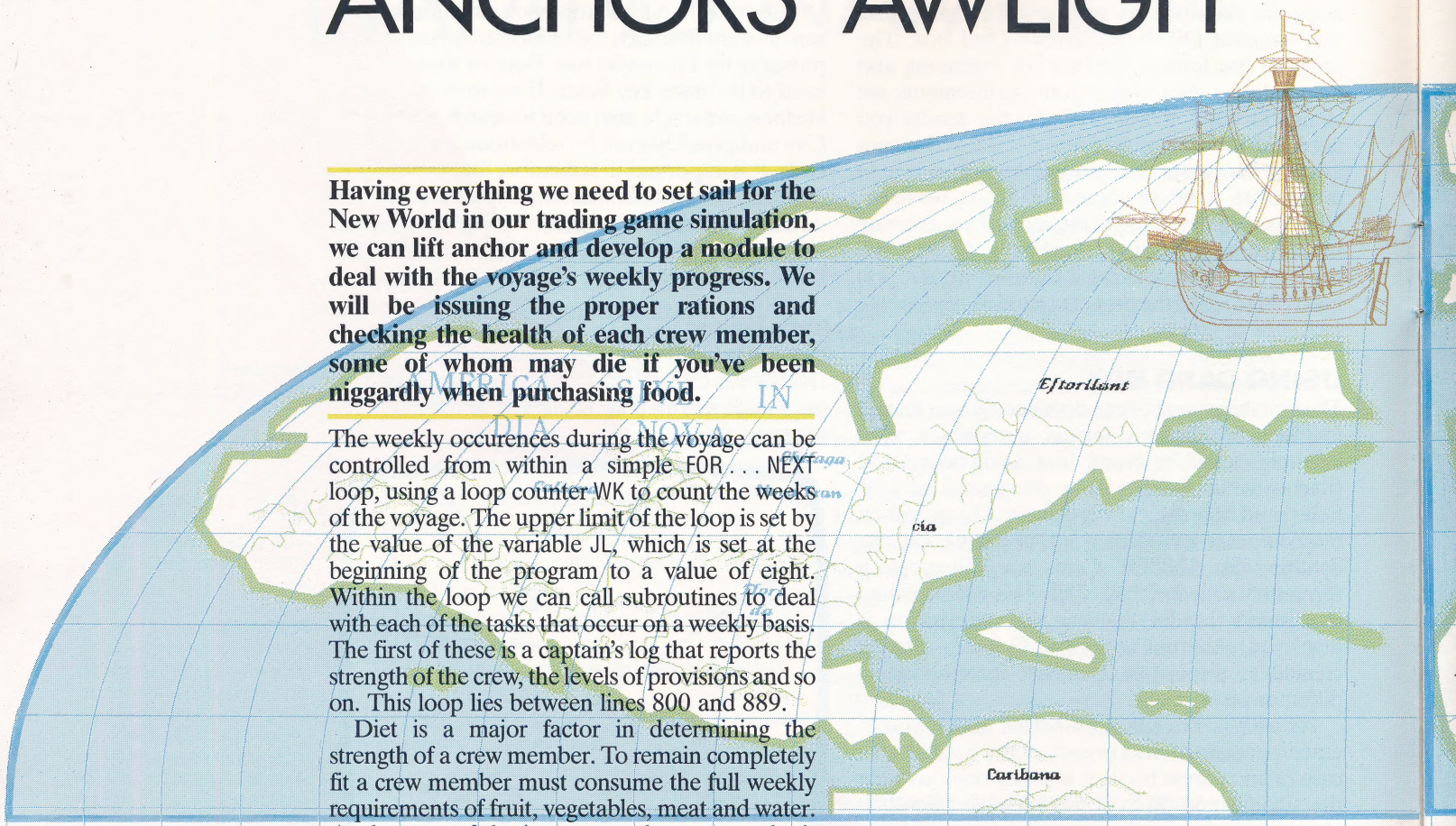
COMMAND LINE
This shows some of the commands that may be used when editing a design. Lines and characters may be inserted and deleted, and the finished design can be saved to disk by pressing CTRL-C



Allow Us To Show You Our Card

Database management systems may offer several different methods of manipulating information. This screen dump of Megafinder, a DBM running on the Apple computer, shows a selected record from a file named 'Business Card' and, below this, a list of command options. Data held by the program can be sorted into up to four different INDEXes to speed information recovery. The DBM will also search for a MATCH between input data and stored data, JUMP to specified sections of the database (e.g. to the section holding company names starting with M), CHANGE and DELETE information, and PRINT a record or report, either to the screen or a printer

ANCHORS AWEIGH



Having everything we need to set sail for the New World in our trading game simulation, we can lift anchor and develop a module to deal with the voyage's weekly progress. We will be issuing the proper rations and checking the health of each crew member, some of whom may die if you've been niggardly when purchasing food.

The weekly occurences during the voyage can be controlled from within a simple FOR . . . NEXT loop, using a loop counter WK to count the weeks of the voyage. The upper limit of the loop is set by the value of the variable JL, which is set at the beginning of the program to a value of eight. Within the loop we can call subroutines to deal with each of the tasks that occur on a weekly basis. The first of these is a captain's log that reports the strength of the crew, the levels of provisions and so on. This loop lies between lines 800 and 889.

Diet is a major factor in determining the strength of a crew member. To remain completely fit a crew member must consume the full weekly requirements of fruit, vegetables, meat and water. At the start of the journey, each crew member's strength is set at 100 in the array TS(,). The program makes a weekly calculation to check if there are sufficient stocks of each provision to last the journey. If supplies of a particular type run low, the player is asked if the crew should go onto half rations of that provision. The half rations will only be issued for one week and would reduce the crew strength by five units.

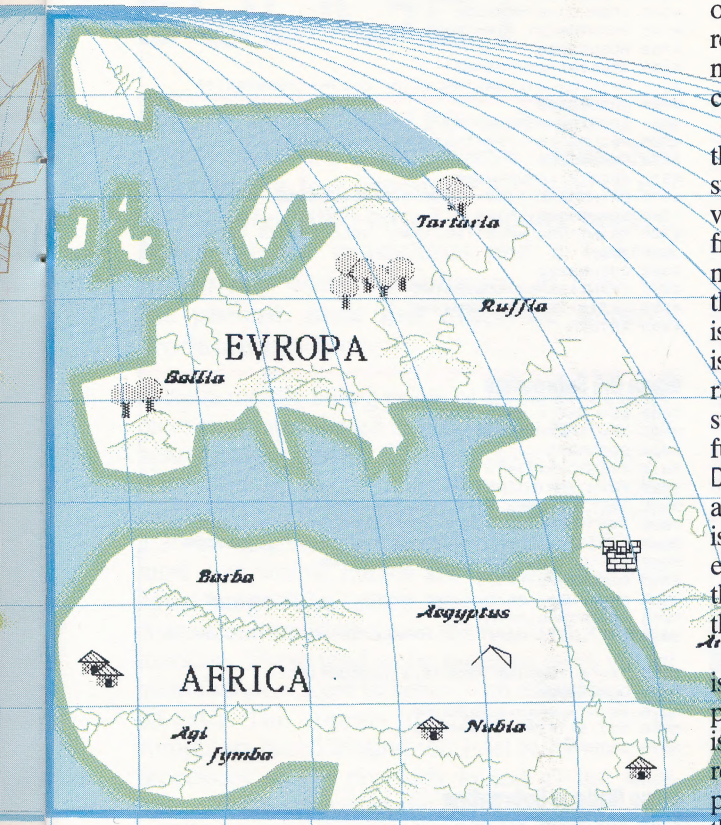
If one type of supply becomes exhausted, each crew member's strength is reduced by 10 for every week they are without it. For example, should the crew run out of fruit and be on half rations of water, their weekly strength rating would decrease by 15. If the water ran out completely, the strength rating would then be reduced by 20 per week. If, after being on half rations for a week, there is enough food or water left to resume full rations for the rest of the trip, they will be automatically issued at the start of the next week.

The health of the crew is given at the beginning of each week by the subroutine at line 4000. There are five classes of health, which are determined by strength rating. The highest classification is very healthy (strength rating 75 – 100), followed by healthy (50 – 75), sick (25 – 50) and very sick (1 – 25). When the strength rating reaches 0, the crew member dies and is buried at sea, but the wages of the deceased are still accounted for each week and paid to the next of kin when the ship returns to port. This subroutine also gives the wage bill for the coming week, the total wages for the voyage so far and the money remaining. The accumulated wage total is reset to zero at the start of this section, at line 800, and is increased by the weekly wage bill each time through the loop. If the bill becomes larger than the balance of gold (as checked by the subroutine at line 5000), the player is warned that a trading profit will be needed to pay the crew.

If all goes well, the voyage should take eight weeks, but unforeseen circumstances could increase the value. Line 801 creates a string variable, H$, to indicate if the crew are on half rations. The routine determines the strength and type of each crew member by checking the type/strength array, TS(,). It sets up a loop at line 4060 for each possible member. There are spaces in the array for information on 16 crew, so the loop is set to 16. If the crew type is recorded as 0, that section of the array will be empty, so line 4070 moves the program along to line 4110. If there is a member of the crew recorded in the array, line 4075 prints the description.

DEATH OF A CREW MEMBER

When a crew member's rating reaches 0, the death must be recorded in the captain's log. As a zero strength rating indicates death, each empty section on the array would appear to represent a dead crew member. To avoid this problem, it is reset to -999 for one week when a rating reaches 0. At the



end of the week it is set back to 0 and the program informs of the death at the start of the next week. Re-setting it to 0 causes it to ignore that rating during the following weeks. This avoids printing a record of dead crew members at the start of each week. The strength of the remaining members is checked by lines 4080 to 4098. Their rating is matched against the four states of health, and line 4099 will print the state of each. The strength/type loop ends at 4110, which sends the program back to look at the next section of the array.

The weekly wage bill is calculated using another loop between lines 4120 and 4135, and used once for each of the five crew types. Wages for each type are calculated by the equation at line 4130, multiplying the count for each type by the wage rate, and accumulating it in the variable, WW. Created and set to 0 at the start of each week by line 4119, WW represents the weekly wage, printed at line 4145. It is then added to the total for the voyage so far (WT), and printed at lines 4160 and 4165. The player can compare this with the remaining money, which is printed at line 4175.

The subroutine at line 5100 handles the issue of rations to the crew. Using a FOR...NEXT loop to repeat the process for each ration type, several checks are made on the level of rations remaining. The first of these is a check to ensure that some rations remain. This can be tested by simply checking that the amount of the provision, held in PA(), is greater than zero. If this is not the case then the strength of each crew member is reduced by a factor of 10. This reduction is actually handled by a small subroutine at line 9300 that reduces the strength rating of each crew member by the value of WF. This subroutine can therefore be used to reduce the crew strength by differing amounts. We need only to set WF to the desired value before calling the subroutine.

If the amount of provisions remaining is greater than zero, then a further check is made to see if sufficient provisions remain for the rest of the voyage. This amount is calculated in line 5180 from the number of crew, CN, the weekly provision needs PN() and the number of weeks remaining in the voyage. If the amount of provisions remaining is less than the projected requirements, the player is given the option of placing the crew on half rations of that particular type of provision. The status of each provision type (whether on half or full rations) is indicated by the array HR() DIMensioned at line 802. When on full rations the array element is set to 1; if the particular provision is reduced to half then the corresponding array element is set to 0.5. If the crew is on half rations then the subroutine at line 9300 is used to decrease the strength rating of each crew member by five.

The final check made by this routine before issuing the ration is to see if the amount of the provision remaining is less than the amount to be issued. If this is the case, then only the amount remaining is issued and, to indicate that no provisions are left, the corresponding element of the provision array is set to -999.

Basic Flavours

Spectrum:
Make the following changes:

```
 847 LET I$ = INKEY$:IF I$ = "" THEN GO TO 847
4010 CLS
4190 LET I$ = INKEY$:IF I$ = "" THEN GO TO 4190
4205 CLS
4295 LET I$ = INKEY$:IF I$ = "" THEN GO TO 4295
4305 CLS
4398 LET I$ = INKEY$:IF I$ = "" THEN GO TO 4398
5010 CLS
5080 LET I$ = INKEY$:IF I$ = "" THEN GO TO 5080
5103 CLS
5220 INPUT I$:LET I$ = I$ (1 TO 1)
5298 LET I$ = INKEY$:IF I$ = "" THEN GO TO 5298
```

BBC Micro:
Make the following changes:

```
4010 CLS
4190 I$ =GET$
4205 CLS
4295 I$ = GET$
4305 CLS
4398 I$ = GET$
5010 CLS
5298 I$=GET$
```

ERRATUM
In the listing of our adventure game Digitaya, on page 1020, BBC Micro users should change all occurrences of the variable names LNS() to LS() and LN to L

Module Four: The Voyage

Length Of Voyage Variable

```
40 JL=8:REM LENGTH OF VOYAGE
```

Main Voyage Loop

```
800 WT=0
801 H$="N" :REM HALF RATION INDICATOR
802 DIMHR(4):HR(1)=1:HR(2)=1:HR(3)=1:HR(4)=1
820 FORWK=1TOJL : REM MAIN VOYAGE LOOP
825 GOSUB4000: REM CREW STATUS REPORT
830 GOSUB4200:REM PROVISIONS REPORT
835 GOSUB4300:REM OTHER GOODS REPORT
840 GOSUB9200:PRINTCHR$(147)
842 PRINT:PRINT:PRINT
843 S$="IT IS ESTIMATED THAT THE VOYAGE*":GOSUB910
0
844 PRINT"WILL TAKE A FURTHER";JL-WK+1;"WEEKS"
845 GOSUB9200
846 PRINT:S$=K$:GOSUB9100
847 GETI$:IFI$=""THEN847
850 GOSUB5000:REM CHECK WAGE BILL
855 GOSUB5100: REM ISSUE RATIONS
889 NEXT WK
```

Crew Status Report

```
4000 REM CREW STATUS REPORT
4010 PRINTCHR$(147)
4020 S$="          CAPTAINS LOG*":GOSUB9100
4025 S$="          -----------*":GOSUB9100
4030 GOSUB9200
4035 PRINT"AT THE START OF WEEK";WK
4040 S$="THE STATE OF THE CREW IS:*":GOSUB9100
4045 GOSUB9200:PRINT
4055 PRINT
4060 FORT=1TO16
4070 IFTS(T,1)=0THEN4110
4075 PRINTC$(TS(T,1));"     (";
4078 IFTS(T,2)=-999THENS$="DEAD !!!!!!!)*":GOTO409
9
4080 IFTS(T,2)>75THENS$="VERY HEALTHY)*":GOTO4099
4085 IFTS(T,2)>50THENS$="HEALTHY)*":GOTO4099
4095 IFTS(T,2)>25THENS$="SICK ! )*":GOTO4099
4098 S$="VERY SICK !! )*"
4099 GOSUB9100:GOSUB9200
4110 NEXT T
4115 GOSUB9200:PRINT
```

Weekly Wage Bill

```
4119 WW=0
4120 FORT=1TO5
4130 WW=WW+(CC(T)*WG(T))
4135 NEXT
4140 S$="WAGE BILL FOR THE WEEK*":GOSUB9100
4145 PRINTWW;"GOLD PIECES"
4150 GOSUB9200
4155 WT=WT+WW
4160 S$="TOTAL WAGES FOR VOYAGE SO FAR:*":GOSUB910
0
4165 PRINTWT;"GOLD PIECES"
4170 GOSUB9200
4175 PRINT"MONEY LEFT = ";MO;"GOLD PIECES"
4180 PRINT:S$=K$:GOSUB9100
4190 GETI$:IFI$=""THEN4190
4199 RETURN
```

Provisions Report

```
4200 REM PROVISIONS REPORT
4205 PRINTCHR$(147)
4206 PRINT"AT THE START OF WEEK";WK:GOSUB9200
4210 S$="YOU HAVE THE FOLLOWING*":GOSUB9100
4215 S$="PROVISIONS LEFT:*":GOSUB9100
4220 PRINT:GOSUB9200
4225 FORT=1TO4
4226 IFPA(T)=0ORPA(T)=-999THEN4240
4230 PRINTPA(T);U$(T);"S OF ";P$(T)
4232 X=INT(PA(T)/(CN*PN(T)))
4235 PRINT"(ENOUGH FOR";X;" WEEKS)"
4239 GOSUB9200
4240 NEXT
4290 PRINT:S$=K$:GOSUB9100
4295 GETI$:IFI$=""THEN4295
4299 RETURN
```

Other Goods Report

```
4300 REM OTHER GOODS REPORT
4305 PRINTCHR$(147)
4306 PRINT"AT THE START OF WEEK";WK:GOSUB9200
4310 S$="YOU ALSO HAVE:*":GOSUB9100
4320 PRINT:GOSUB9200
4322 IFOA(1)=0THEN4332
4325 PRINTOA(1);:S$="BOTTLES OF MEDICINE*":GOSUB91
00
4330 GOSUB9200
4332 IFOA(2)=0THEN4342
4335 PRINTOA(2);:S$="GUNS*":GOSUB9100
4340 GOSUB9200
4342 IFOA(3)=0THEN4352
4345 PRINTOA(3);:S$="BAGS OF SALT*":GOSUB9100
4350 GOSUB9200
4352 IFOA(4)=0THEN4362
4355 PRINTOA(4);:S$="BALES OF CLOTH*":GOSUB9100
4360 GOSUB9200
4362 IFOA(5)=0THEN4372
4365 PRINTOA(5);:S$="KNIVES*":GOSUB9100
4370 GOSUB9200
4372 IFOA(6)=0THEN4380
4375 PRINTOA(6);:S$="JEWELS*":GOSUB9100
4380 GOSUB9200:PRINT
4382 PRINT"YOU HAVE";MO;:S$=" GOLD PIECES LEFT*":G
OSUB9100
4384 GOSUB9200
4397 PRINT:S$=K$:GOSUB9100
4398 GETI$:IFI$=""THEN4398
4399 RETURN
```

Wage Bill Subroutine

```
5000 REM CHECK WAGE BILL
5005 IFWT>MOTHEN5010
5008 GOTO5099
5010 PRINTCHR$(147)
5020 PRINT:PRINT:PRINT
5025 S$="THE CREW HAVE HEARD A RUMOUR*":GOSUB9100
5030 S$="THAT YOU DON'T HAVE ENOUGH*":GOSUB9100
5035 S$="GOLD TO PAY THEM AT THE END*":GOSUB9100
5040 S$="OF THE VOYAGE.*":GOSUB9100
5045 GOSUB9200:PRINT
5050 S$="THEY ARE GETTING ANGRY !!*":GOSUB9100
5055 GOSUB9200:PRINT
5060 S$="LET'S HOPE YOU MANAGE TO MAKE*":GOSUB9100

5065 S$="A TRADING PROFIT!*":GOSUB9100
5066 GOSUB9200
5070 PRINT:S$=K$:GOSUB9100
5080 GETI$:IFI$=""THEN5080
5099 RETURN
```

Issue Rations Subroutine

```
5100 REM ISSUE RATIONS
5103 PRINTCHR$(147)
5105 S$="       ISSUING RATIONS*":GOSUB9100
5106 S$="       ---------------*":GOSUB9100
5107 GOSUB9200:PRINT"WEEK";WK:PRINT
5108 H$="N"
5110 FORT=1TO4
5112 HR(T)=1
5115 IFPA(T)>0THEN5180
5120 PRINT"NO ";P$(T);" LEFT!!!":GOSUB9200
5130 S$="THE CREW IS GETTING WEAKER !!*":GOSUB9100

5135 WF=10:GOSUB9300
5139 GOTO5290
5180 X=(PN(T)*CN)*(JL-WK+1)
5185 IFPA(T)<XTHEN5200
5190 GOTO5270
5200 PRINT"RUNNING SHORT OF ";P$(T)
5205 GOSUB9200
5210 S$="DO YOU WANT TO PUT THE CREW ON*":GOSUB910
0
5215 PRINT"HALF RATIONS OF ";P$(T)
5220 INPUTI$:I$=LEFT$(I$,1)
5221 IF I$<>"Y"AND I$<>"N" THEN 5220:REM INPUT ERR
OR
5225 IF I$="N"THEN 5270
5230 HR(T)=.5:H$="Y"
5240 WF=5:GOSUB9300
5250 S$="THE CREW IS GETTING WEAKER!*":GOSUB9100
5270 X=PN(T)*HR(T)*CN
5272 IFX>PA(T)THENX=PA(T)
5275 PA(T)=PA(T)-X
5280 IFPA(T)=0THENPA(T)=-999
5285 PRINTX;U$(T);"S OF ";P$(T);" ISSUED"
5290 PRINT:GOSUB9200:NEXT
5295 PRINT:S$=K$:GOSUB9100
5298 GETI$:IFI$=""THEN5298
5299 RETURN
```

Reducing Crew Strength

```
9300 REM REDUCE CREW STRENGTHS BY WF
9310 FORS1=1TO16
9320 TS(S1,2)=TS(S1,2)-WF
9330 IFTS(S1,2)<1THENTS(S1,2)=-999
9340 NEXT
9399 RETURN
```

BUSINESS ON A BUDGET

Similar to the IBM PC and priced at under £1,000, the Sanyo MBC-550 represents a less expensive addition to the 16-bit Intel 8088-based computer market. But hardware restrictions and a lack of specifically designed software, at least for the present, limit this otherwise sensibly produced machine.

Despite rapidly falling prices of 16-bit processors, most computers based around 16-bit technology, apart from the Sinclair QL, are still very expensive. It is hard to understand why this is. Sinclair Research had proved that it is possible to produce a 16-bit machine profitably for under £400, yet most comparable micros are well into the four-figure range. The reason probably lies with the type of customer who is in the market for such a machine. These tend to be business users whom manufacturers seem to believe can afford to pay a bit extra. This appears to be particularly true of computers based around the Intel 8088 series of processors and is probably linked to the sales policy of the market leader, IBM.

With the standard IBM PC including a single disk drive costing around £2,000, most manufacturers building 8088-based micros consider that by shaving a few hundred pounds off this price, their customers will regard that as a bargain. But this still leaves a section of the market that cannot quite afford many of these, although still requiring a computer for business use. Until recently, this area of the market was occupied by business computers based around the Z80 processor that were equipped with disk drives enabling them to run CP/M. However, many manufacturers are now realising the potential of this neglected area of the business market, and are endeavouring to produce inexpensive 8088-based machines.

The Sanyo MBC-550 is one of the first of these to arrive on the market. Although Sanyo does not claim it to be compatible with the IBM PC, the MBC-550 does run the *de facto* standard MS-DOS disk filing system. The machine is available in either a single or twin disk version and sells for around £1,000.

Like most business machines, the MBC-550 has two separate units, comprising the keyboard and the main computer. These are sturdily cased in a combination of metal and plastic, finished in a metallic silver. The keyboard is divided into three sections. On the far left are five programmable function keys which, when used in conjunction with the Shift key, produce a maximum of 10 functions that vary according to which application is currently in use. In BASIC, these keys can be used as single keyword entries with commonly used commands such as LIST, LOAD and RUN.

The typewriter keyboard is well laid out with all the control keys in their usual positions. The control keys include a destructive Backspace and an Insert/Delete key, only one of which is operational in each application. There is also a very large Return key, four times the size of one of the normal keys. On the right-hand side of the Space bar is a Graphics key, which produces graphics characters on the screen when locked. On the far right-hand side of the keyboard is a numeric keypad that can be used as a calculator, or, if the Number Lock key is pressed, as a cursor control to enable full screen editing in BASIC.

Sibling Differences
The Sanyo MBC-550 is a budget priced 16-bit business machine which runs under the popular MS-DOS operating system. There are two versions of the machine: the MBC-550 is the single drive version and the dual drive configuration is known as the MBC-555. The monitor shown here is not included in the basic price of around £1,000

CHRIS STEVENS

The computer itself is a large flat box about the size of a video recorder, that has been made large enough to accommodate the optional Sanyo monitor. Like the keyboard, it is mostly manufactured of sheet metal. On the front of the computer is the power switch and space for two disk drives, although our illustration shows the single drive model. The drives used by the MBC-550 are of the standard 5¼in floppy disk variety. On the front of each drive is a clip to hold the disk in position and prevent it from being removed while the drive head is reading it.

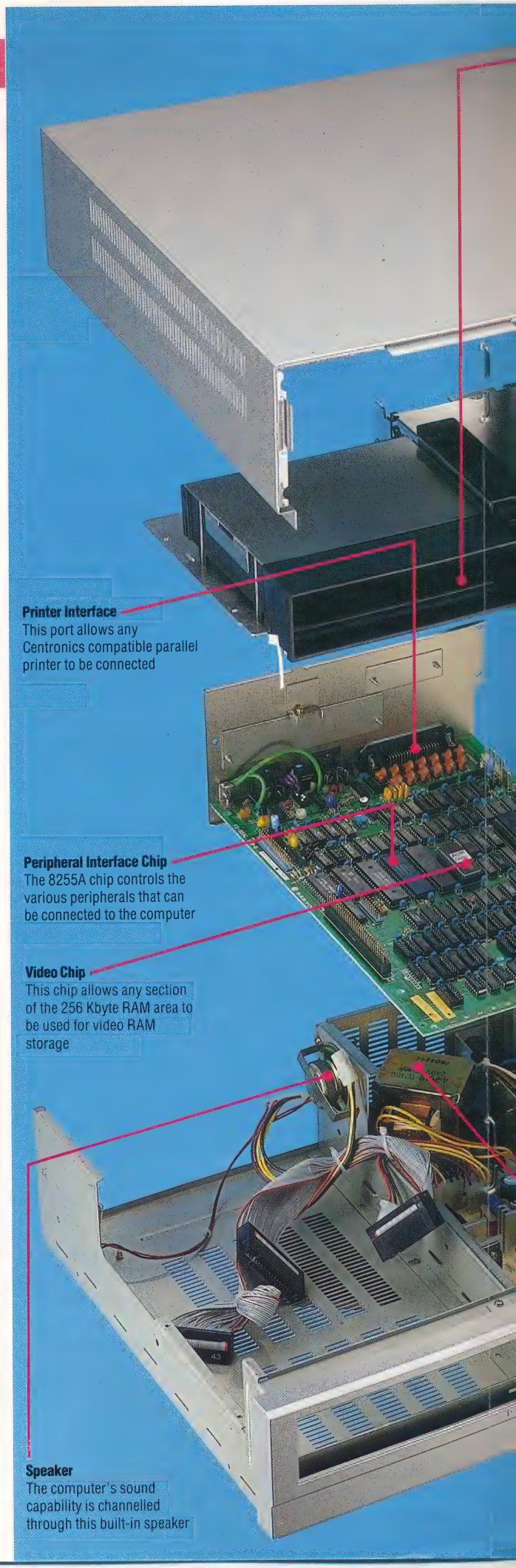
PERIPHERAL INTERFACES

The rear of the computer contains the power supply and the peripheral interfaces. Looking at the back of the computer, on the left side, you will see the power cable, the fuse housing and the earth connection. On the right, there is a Centronics parallel interface that will accommodate the connecting of a printer. A pair of monitor sockets are to the right of this. The first is an RGB socket, enabling the computer to run a colour monitor, while the other is a composite video jack that plugs into the Sanyo green monochrome monitor. Above these interfaces are further expansion ports provided to accommodate extra peripheral interfaces at a later date.

The Line port above the printer interface is included for the connection of an RS232C serial interface that would facilitate the computer communicating with other computers, via a modem, or with any other serial device, such as a printer. Next to that is a port in which an MBC series or Apple-compatible joystick interface could be fitted. This not only allows a joystick to be fitted but would also enable the computer to be controlled from a trackball, paddle or other external device. There is also an external bus socket allowing other peripheral devices to be interfaced.

Unlike many manufacturers who object strongly to users tampering with their machines, Sanyo has thoughtfully included, in the user manual, instructions for the fitting of all of these peripheral interfaces, extra banks of RAM and a second disk drive if only one is currently fitted. This is a very welcome inclusion, although the guide does mention that users should consult a qualified engineer if they are at all uncertain. The rest of the user manual is equally helpful. It contains a glossary of computer terms, a full

Room To Expand
Although the MBC-550 is not equipped with as many peripheral options as some of the more expensive machines that are currently available, Sanyo have designed the machine to be expandable. There are extra slots to allow interfaces for joysticks and serial devices, and fitting instructions are provided in the user's manual

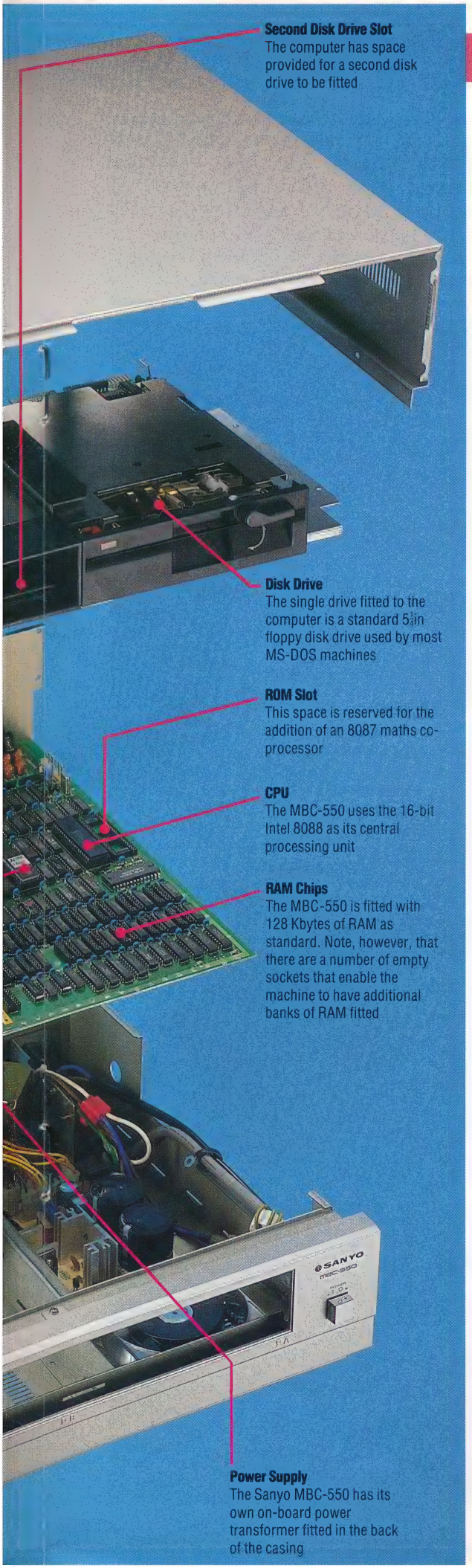


Printer Interface
This port allows any Centronics compatible parallel printer to be connected

Peripheral Interface Chip
The 8255A chip controls the various peripherals that can be connected to the computer

Video Chip
This chip allows any section of the 256 Kbyte RAM area to be used for video RAM storage

Speaker
The computer's sound capability is channelled through this built-in speaker



tutorial description of Sanyo BASIC — including instructions on using the disk system from BASIC — and a wealth of technical information.

Sanyo BASIC, a derivative of Microsoft's version, is certainly adequate, although it seems to be somewhat slow in processing arithmetical functions. The 8088 chip has long been noted as being sluggish when processing maths (hence the popularity of the 8087 maths co-processor), but the MBC still seems noticeably slower than most other 8088-based machines. Similarly, line drawing on the MBC-550 is also very slow. This is because Sanyo BASIC lacks a DRAW command, requiring line drawing to be performed by the PSET command, which simply turns on one specified pixel to a particular colour. Thus, drawing a line involves using a loop.

Apart from standard Microsoft-type commands such as MID$, LLIST and CIRCLE, there are also commands to define and view windows on the screen. Structured programming is encouraged by the inclusion of the WHILE. .WEND conditional statement. Many Sanyo BASIC keywords can be entered by two or three keystrokes based around the Control key. For example, the keyword DIM can be entered by pressing CTRL, SHIFT and D simultaneously, while PRINT is entered by pressing CTRL and P. Although any move towards the simpler entry of BASIC keywords is to be welcomed, and many of the shortened keypress commands bear a relationship to the originals, it is difficult to imagine anyone attempting to remember the 40 commands available to write BASIC programs. More likely, they would attempt to memorise a few of the more commonly used shorthand versions.

Bundled with the Sanyo MBC-550 is an MS-DOS system disk, which also contains the Sanyo BASIC, the widely used WordStar word processing package and the CalcStar spreadsheet. Accompanying these packages is a manual giving full descriptions on how to use them.

One would expect the standard MBC-550 — based around the 8088 chip and running under MS-DOS — to have little problem running the vast amount of IBM PC software that is available. But the fact is that hardware restrictions within the machine meant that none of the IBM-compatible packages that we tested on the computer would run.

There is undoubtedly a market for low-priced MS-DOS business machines able to run IBM-compatible software. The question is not whether they will appear, but when. The Sanyo MBC-550 is a first attempt to break the £1,000 barrier, but unfortunately its lack of compatibility means that it will remain isolated. For the small business owner who only needs a computer to run competent word processing and spreadsheet facilities, the MBC-550 seems to be good value. However, anyone wishing to gain access to a wider software base than is available for the Sanyo will have to reconcile themselves to either spending quite a bit more, or else waiting a little longer.

SANYO MBC-550

PRICE
Standard MBC-550: £861.
Twin version MBC-555: £1,149.
Both inc VAT

DIMENSIONS
375x355x108mm

CPU
Intel 8088 running at 3.6 MHz

SCREEN
80x25 text screen, giving 640x200 pixels high resolution mode. There are up to eight colours available in high resolution

INTERFACES
Centronics parallel port, with options for an RS232 interface to be fitted, and an Apple-compatible joystick port

LANGUAGES AVAILABLE
Sanyo BASIC

KEYBOARD
65 typewriter keys, five programmable dual function keys and a 19-key numeric keypad

DOCUMENTATION
The manual is very good, containing a full explanation of the BASIC keywords, although not, unfortunately, a tutorial course. There is also an introduction to MS-DOS, WordStar and CalcStar. Unusually for a business machine, the user manual includes a wealth of technical information including instructions enabling users to fit their own expansion boards

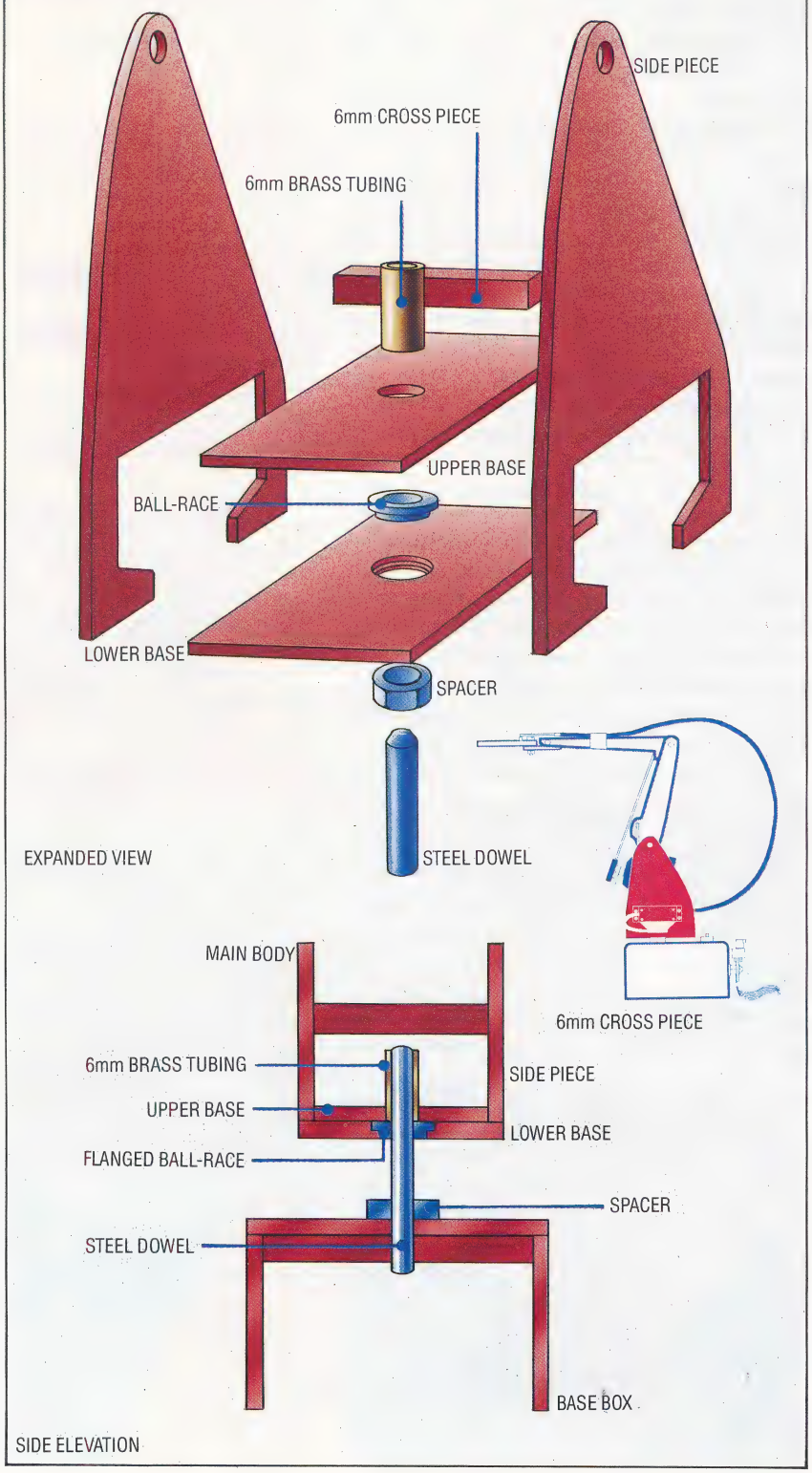
STRENGTHS
At the price, the Sanyo MBC-550 certainly looks a good buy, providing full 16-bit business computing at a fraction of the normal price

WEAKNESSES
Because the machine lacks compatibility with most existing software, the computer is faced with the usual problem encountered by new machines — the lack of programs available

THE ARM'S BUILD-UP

After listing the necessary parts for our Workshop robot in the previous instalments, we can now begin its construction. Here we assemble the main body, the upper and lower arm sections, as well as make the elbow and shoulder joints using the previously cut out parts.

Step 1: Main Body Assembly



Step 1: Main Body Assembly

Drill a hole in the lower base piece (the wider of the two base pieces) in order to accept the ball-race. Using a larger size drill or a countersinker, bevel the upper lip of the hole to seat the ball-race flange, as shown. Then, drill a 6mm diameter hole in the upper base piece to take in the 25mm length of 6mm internal diameter tubing. With the holes aligned and the ball-race seated in the lower base piece, glue the two base pieces together. Insert the brass tubing into the hole from the top and glue into position.

When this assembly is dry the main body side pieces can be added. Drill out the holes at the top of the side pieces so that they will snuggly accept a piece of 4mm external diameter tubing — this should be a friction fit. Insert a 37mm length of 6 x 6mm wood as a cross piece between the two sides — positioned at the rear of the assembly in line with the motor cutout in the side piece — and glue the assembly together. When the assembly is dry, place a spacer (say, a large nut) on the steel dowel that protrudes from the top ot the base box, which we assembled on page 1094. Push the dowel through the ball-race and check that the main body assembly can rotate freely.

Step 2: Lower Arm Assembly

Cut a 37mm length of 5mm external diameter brass tubing and drill holes in the middle of the lower arm pieces to take in this tubing. Make sure that you achieve a snug fit, as this brass tubing will form part of the shoulder joint. Drill two further holes at the tapered end of the lower arm pieces to take in the 4mm external diameter brass tubing. As before, make sure that a friction fit is obtained.

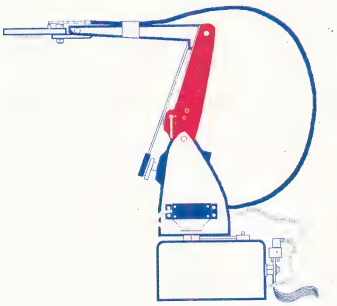
Working at the middle, thread the 5mm tubing through the holes in the lower arm pieces using one of the servo motors as a spacer as shown. Cut two 20mm lengths of 6mm wood as cross pieces and fit them between the lower arm pieces. These locate just under the brass tubing and at the bottom of the assembly, and are used to mount the servo motor. Glue these in place and, when they've dried, mount the motor using four self-tapping screws and rubber grommets. The lower arm assembly can now be attached to the main body at the shoulder joint. Now, insert a 45mm length of 4mm tubing through one of the holes at the top of the main body and through the brass tubing already in place in the lower arm assembly.

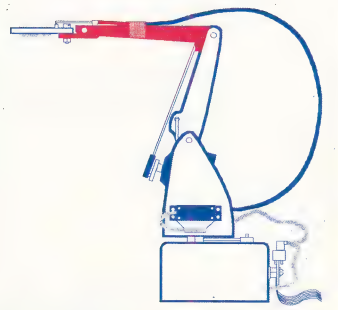
Make sure that the lower arm assembly is centred, sliding it on the brass pivot if necessary. Mark the position when it seems properly aligned, disassemble, and then glue the larger brass tube in place on the lower arm assembly. Reassemble the shoulder joint when dry.

Step 3: Upper Arm Assembly

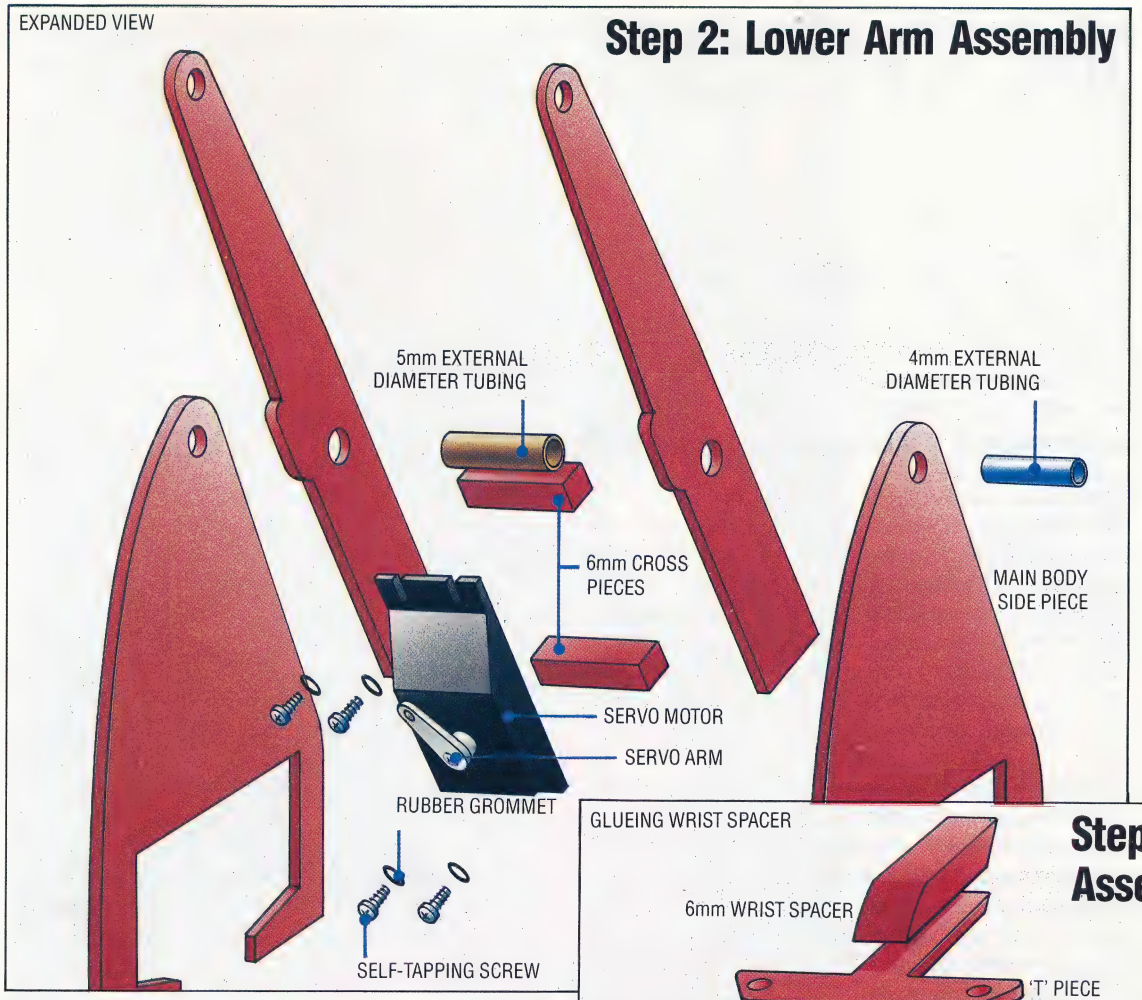
Drill out two holes to take in the 5mm brass tubing at the wider ends of the upper arm pieces. Drill a further two holes at the other ends, large enough to take a machine screw. Glue a 25mm length of 6 x 6mm wood onto the 'T' piece as shown and shape the ends. This will act as a wrist spacer. Place the 'T' section between the two upper arm pieces and drill through the wrist spacer so that a machine screw can be pushed through the spacer and upper arm sections. Fix the 'T' piece in place with a washer and a nut.

At the other end of the arm push a 17mm length of 5mm brass tubing through the drilled holes, and connect the upper arm assembly to the lower arm at the elbow joint by threading a 25mm length of 4mm tubing through the joint as before. Again, check that the upper arm is central before disassembling and gluing the larger diameter brass tubing in place on the upper arm.

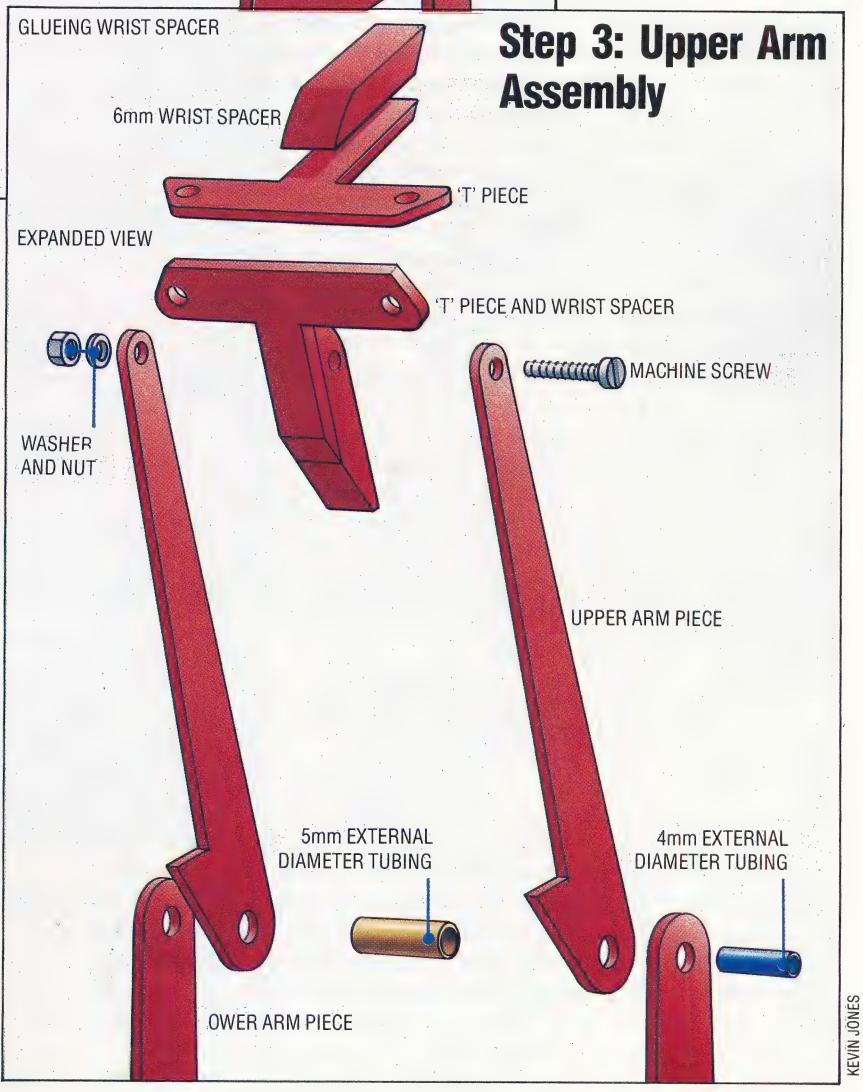


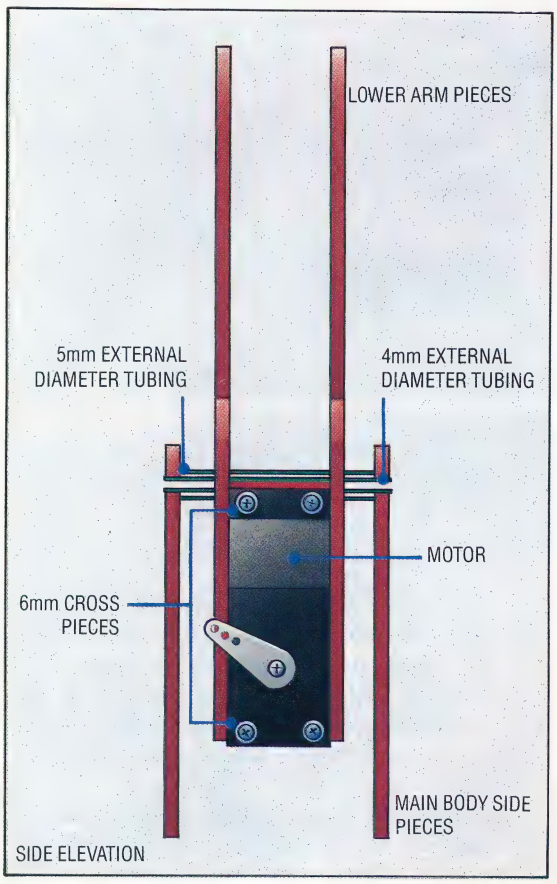


Step 2: Lower Arm Assembly



Step 3: Upper Arm Assembly





KEVIN JONES

ON THE SET

In addition to PASCAL's familiar data structures, such as arrays and files, the language also includes sets and records. We examine the first of these structured types, as well as summarising the levels of precedence for the PASCAL operators we have examined so far.

We often talk about a computer's character set. But what precisely is meant by a set and how does it differ from an array? The main characteristic of a set is that it is a collection of objects that can be processed as a single entity, rather than each element having to be accessed individually. A practical example would be the set of all people who program in PASCAL, or the letters of the alphabet that are vowels. There is often no particular ordering involved, with the only question of interest being: is this particular object a member of the set or not?

For reasons of efficient implementation, PASCAL places some restrictions on sets. They can only have simple scalar types as their members — not arrays or other structured data — and there will be an (implementation-defined) upper limit on the range allowed for this 'base' type. The syntax is straightforward, for example:

```
TYPE
   Numbers    = SET OF 0 . . 127;
   Alphabet   = SET OF 'A' . . 'Z';
   ColourMix  = SET OF ( Red, Green, Blue );
```

The range of possible values of the ordinal base type of the set may be expressed with the same syntax as for subrange types.

Set variables are declared in the VAR declaration part, in exactly the same way as all variables in PASCAL, and may appear in statements literally (i.e: as 'literals') enclosed within square brackets, thus:

```
VAR
   codes  : Numbers;
   palette  : ColourMix;
BEGIN
   codes := [0 . . 2, 4, 8, 16, 32, 64 ];
   palette := [Red . . Blue ]; { etc. }
```

This segment would initialise the set codes to contain only the numbers 0 to 2 inclusive and 4, 8 and so on, as listed. Because there is no inherent order in the set itself (as opposed to its base type), we could equally well express this set as [64, 32, 16, 8, 4, 0 . . 2] but notice that the subrange 2 . . 0 (illegal in an actual subrange definition) would merely indicate an empty range. In fact, any set can be initialised to an empty set of its type with the statement: AnySet :=[ ], which gives rise to the only exception to the general rule in PASCAL, namely, the type of any literal is known by inspection. Without at least one member of the set appearing literally, neither we nor the compiler can determine its type. Fortunately, the empty set can only occur in assignments (as in the example) or expressions in which the other identifiers will have already had their types declared. This does mean, however, that the empty set is a subset of every set type, but that's only natural.

SET OPERATORS

One of the most useful operators PASCAL provides for set structures is, like DIV and MOD, a reserved word: IN. It enables us to test for membership of a set, and is a relational operator that takes two operands. The left-hand side must be an expression evaluating to one of the possible members of the set — in other words, a value of the set's base type — and the right-hand side may be a set variable or set literal. The whole expression will give a Boolean result — true if the value is a member of the set and false otherwise.

So: N IN codes and Green IN palette are legal Boolean expressions. The usual test for a character being a digit could be written as:

```
IF (c >= '0' ) AND (c <= '9' ) THEN . . .
```

How much less confusing to test for c's value being a member of the set of characters we're interested in, using:

```
IF c IN ( '0' . . '9' ) THEN  . . .
```

Or perhaps, if we are writing a program to play a card game:

```
TYPE
   rank =(deuce, three, four, five, six, seven, eight,
          nine, ten, Jack, Queen, King, Ace );
   CardSet = SET OF rank;
VAR
   card  : rank;
   pictures: CardSet;
BEGIN
   pictures :=( Jack . . Ace );
   IF card IN pictures THEN { etc. }
```

Contrast this last example with:

```
IF (card = Jack) OR (card = Queen) OR (card = King)
OR . . .
```

There are also operations defined on sets as a whole in PASCAL. These are set intersection, union and difference. If B is the set of all BASIC programmers and P represents PASCAL people, then the intersection of the two sets is the set of programmers using *both* BASIC and PASCAL. The union of P and B is the set of people who program in either PASCAL or BASIC — that is, the *combination* of the two sets.

Set difference is, as it suggests, the result of removing or subtracting all members of one set from the other. Thus, in PASCAL notation,

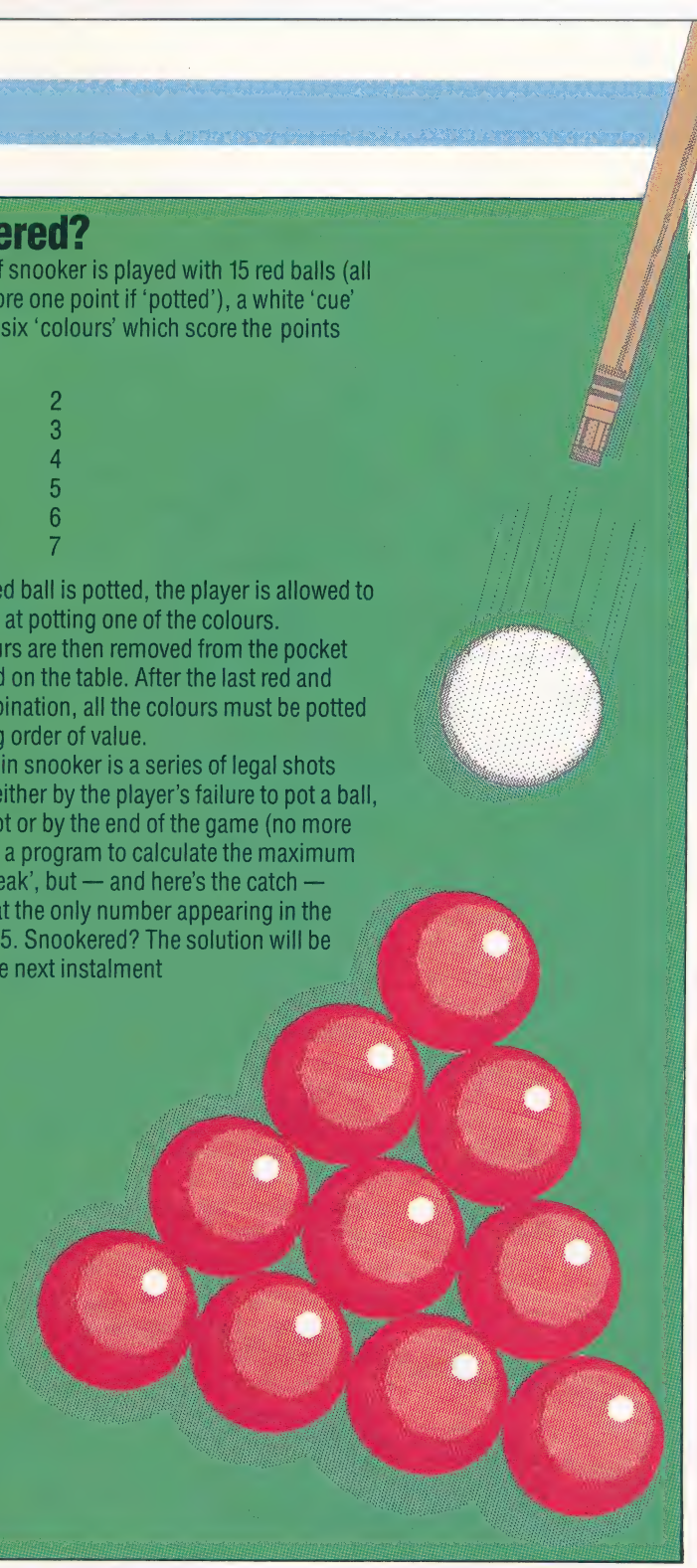
Snookered?

The game of snooker is played with 15 red balls (all of which score one point if 'potted'), a white 'cue' ball and the six 'colours' which score the points shown:

yellow	2
green	3
brown	4
blue	5
pink	6
black	7

After each red ball is potted, the player is allowed to try his hand at potting one of the colours. Potted colours are then removed from the pocket and replaced on the table. After the last red and colour combination, all the colours must be potted in ascending order of value.

A 'break' in snooker is a series of legal shots terminated either by the player's failure to pot a ball, by a foul shot or by the end of the game (no more balls). Write a program to calculate the maximum possible 'break', but — and here's the catch — ensuring that the only number appearing in the program is 15. Snookered? The solution will be printed in the next instalment



IAN McKINNELL

Operator Precedence

The table below shows the order of precedence of PASCAL operators. The use of round brackets forces an enclosed expression to be evaluated separately, hence overriding the normal order of precedence (shown here) if required

Precedence	Operators	Type
Highest	NOT + -	{unary}
	AND * / DIV MOD	{multiplying}
	OR + -	{adding}
Lowest	< <= <> >= > IN	{relational}

P — B represents all those who program in PASCAL but not in BASIC. Similarly, the notation used for union is P + B, and for intersection P * B. These operator symbols happen to be the same as the well-known arithmetic operators, but the different class of operations should not be confused. Actually, the reason they appear so naturally in the context of set operations is that they represent the bit testing that is involved here.

Imagine a set of eight elements. The presence of a particular member could be indicated by setting the appropriate bit in an 8-bit pattern (one byte); absence from the set can be set similarly by a zero. Membership tests, then, just require a mask or bit test, a set union becomes an OR operation and intersection will be a simple AND. These operations are invariably available at machine code level, so that the PASCAL compiler is able to provide set data structures and their associated operations very efficiently. The memory overhead is also minimal, and especially when the size of sets can be kept within a computer's word size (on 32-bit and over machines, for example), set operations can be the most efficiently implemented ones in PASCAL.

ORDER OF PRECEDENCE

We can now conveniently summarise all the operators in PASCAL. Those we haven't dealt with explicitly are all familiar from other languages, and PASCAL keeps things extremely simple by only having four levels of operator precedence. Naturally, the 'unary' or 'monadic' operators take precedence over all the others. These are the symbols + and —, which indicate the sign of a number and the Boolean negation operator NOT. The second precedence level has all the 'multiplication' operators (including the division symbols) followed by addition/subtraction and, at the lowest level of precedence, all the relational operators including IN.

Notice that the other two Boolean operators (AND and OR) are treated correctly as multiplying and adding operators, respectively. This faithfully echoes the actual Boolean algebra involved, and means that many relational tests are bracketed to override this precedence. Otherwise, for instance: IF N > 0 AND N < 10 THEN . . will give a compile time error, because the expression 0 AND N (which should be evaluated first) attempts to combine two integer operands with a Boolean operator.

Perhaps this is another habit you've acquired from BASIC? If so, now is the time to start breaking it! Naturally, any operators of the same precedence level are evaluated from left to right, as usual. The symbol used for assignment (:=) has a lower precedence than any of the above operators, as the expression on the right-hand side of an assignment must be fully evaluated before the assignment is made. One word of caution: you can never assume that any part of an expression won't be evaluated, so that: IF (N > 0) AND (K/N < 10) THEN . . . could crash a program if N was zero (creating a division by zero error !).

Bingo!!

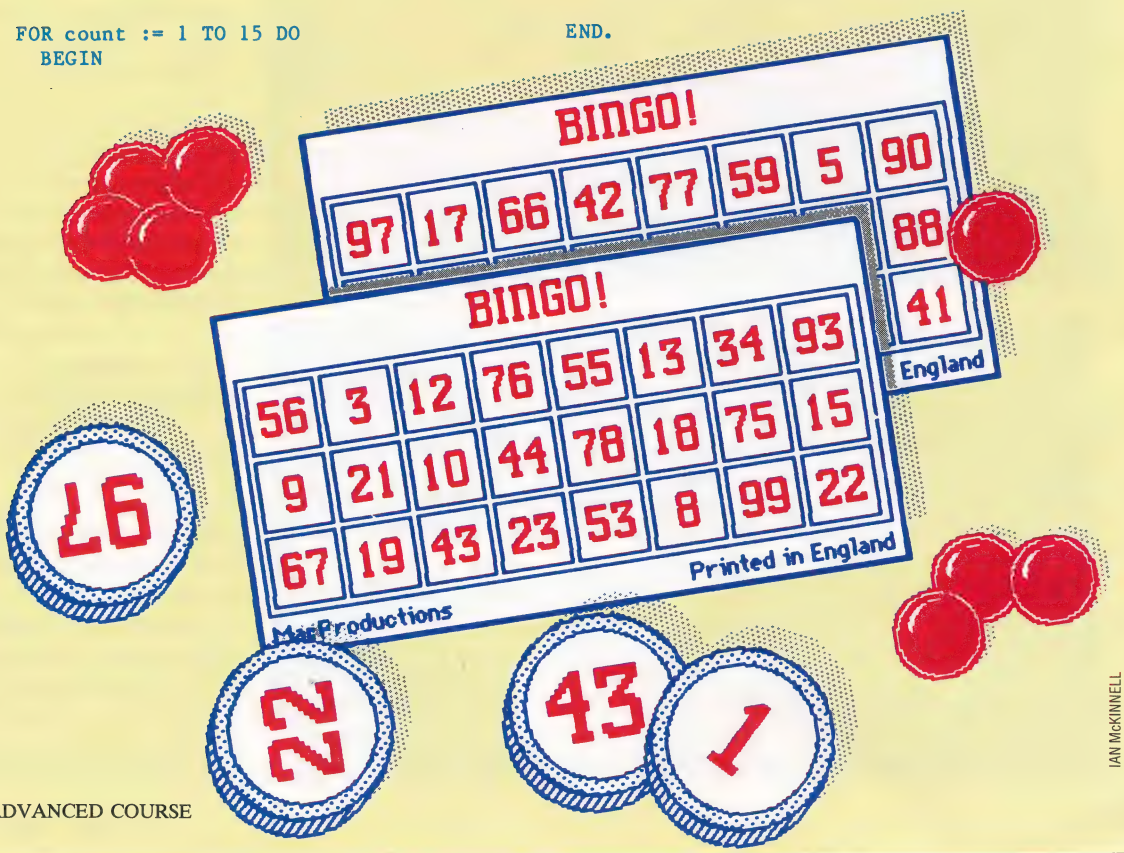
A Bingo card may best be represented by a set. Although the base elements (integers in the range one to 90) are ordered, the only vital consideration is whether or not a called number is on the card. In PASCAL parlance: 'number IN Card' is either true or false.

The program simulates a game of Bingo by first reading the card numbers from the keyboard and then, as each number is 'called', testing for the set of called numbers being a superset of the card. The expression Card—Called will become the empty set when every member of Card is also in the set Called.

```
PROGRAM        Bingo   ( input, output );

CONST
        Columns = 40;   {  to suit VDU  }
        HalfWay = 25;   {       "       }

TYPE
        Bingo   = SET OF 1 .. 90;

VAR
        count   : 1 .. 15;
        Allowed,
        Called,
        Empty,
        Card    : Bingo;
        number  : integer;
        House   : boolean;

BEGIN

  Empty := [ ];
  Allowed := [ 1 .. 90 ];
  WriteLn ( '*** BINGO ***' : HalfWay );
  WriteLn;
  WriteLn ( 'Enter the 15 card numbers,' );
  WriteLn ( '(each followed by RETURN) :' );
  Card := Empty;

  FOR count := 1 TO 15 DO
    BEGIN
      write ( count : 10, ': ? ' );
      ReadLn ( number );
      Card := Card + [ number ]
    END;

  WriteLn;
  WriteLn ( 'EYES DOWN !' : HalfWay );
  WriteLn;
  WriteLn ( 'Now call each number :' );
  Called := Empty;

  REPEAT
    write ( '? ' : HalfWay );
    ReadLn ( number );
    Called := Called + [ number ];
    House := Card - Called = Empty

  UNTIL House;

  WriteLn;
  WriteLn ( 'Congratulations !' : HalfWay );
  WriteLn ( 'Your numbers were :' );
  WriteLn;

  FOR number := 1 TO 90 DO
    IF number IN Card THEN
      write ( number : Columns DIV 8 )

END.
```

Adding a member to a set that already contains it leaves the set unchanged, as does 'removing' a non-member. Notice that we cannot add a member to a set directly; but by creating a one-element set by enclosing the number within square brackets, we can obtain the union of two sets.

As it stands, the program will allow duplicated card entries, and will accept an illegal number outside the range one to 90, causing a run-time error. As an exercise, can you think of a simple way of adding loop constructs to prevent both these anomalies and to reject numbers that have already been called?



IAN McKINNELL

LAST CALL

Our extended look at the BBC Micro's operating system has shown how the various ROM routines that make up the OS can be used by the programmer to augment or totally alter the way in which the OS responds to certain situations. Here, we round up the series with several programs that illustrate the uses of OS calls and point to some of their limitations.

The program we gave as an exercise in the previous instalment (see page 1100), demonstrates how we can intercept one of the OS vectors in order to change the way in which the OS responds to a certain event. The program alters the contents of OSWRCHV — the vector that holds the address of the OSWRCHV routine. This vector is held at addresses &20E and &20F.

One of the functions of OSWRCH is to take BASIC text and print it to the screen when a program is listed. The main function of the machine code in our program is to look at the ASCII code of each character as it passes through the accumulator during the OSWRCH routine. If the ASCII code is between 64 and 91 (the character is a capital letter), then 32 is added to the value to give its lower case equivalent. In this way, the program actually forms a 'wedge' that will be executed each time the operating system calls OSWRCH. After our code has been executed, control is passed to the real OSWRCH routine using a JMP instruction with the old contents of OSWRCHV. After running the program, all capital letters in a program listed to the screen or typed in at the keyboard will be converted to lower case letters by our wedge routine. Hitting BREAK restores OSWRCHV to its normal value.

MEMORY USE PROGRAM

This listing is another utility program, which keeps an eye on the use of memory while you're programming. The expression:

```
HIMEM—(?2+256*?3)
```

gives the amount of memory remaining after the program and variables have taken up their share (locations 2 and 3 hold the address of the top of the BASIC variables table). The expression thus indicates how much memory is left for you to work with. However, continuously PRINTing this value can be a little tedious. The routine, therefore, uses the Character enters input buffer event (see page 1099) to evaluate this expression each time a key is pressed, and sounds a 'beep' when the remaining BASIC memory falls below a certain level. The program is triggered by a keypress event and will continue to operate in the background while you type in your program.

The first task of the program is for the user to determine the threshold at which he wants to be reminded of failing memory resources. This is done by PROCselect-memory, which sets up the threshold in lo-byte/hi-byte form. The program is located at address &0A00, and — as usual for programs that use events — we must first save the contents of all the registers by pushing them onto the stack. After the subroutine that carries out this job is called, the registers are restored and an RTS is executed to return the control of the program to where it was when the event occurred.

The main subroutine calls a further two subroutines to evaluate the amount of memory remaining. The inc routine takes the value stored in locations 2 and 3 and adds the lo-byte and hi-byte of the threshold to it, storing the result in addresses &70 and &71. The sub routine then carries out a 16-

Use Of Memory

```
100 REM **** Using Key Pressed Event ****
110 REM **** To keep an eye on memory use ****
120 MODE 1
130 PROCselect_memory
140 PROCassemble
150 END
160
170 DEFPROCselect_memory
180 INPUT"Beep when how many K left",n:n=n*1024
190 hibyte=n DIV 256
200 lobyte=n MOD 256
210 ENDPROC
220
230 DEFPROCassemble
240 oswrch=&FFEE
250
260
270 FOR pass=0 TO 3 STEP 3
280 P%=&0A00
290
300 [OPT pass
310 .start
320 PHP
330 PHA
340 TXA:PHA
350 TYA:PHA
360
370 JSR memory_check
380
390
400 PLA:TAY
410 PLA:TAX
420 PLA
430 PLP
440 RTS
450
460 .memory_check
470 JSR inc
480 JSR sub
490 BPL room
500 LDA #7
510 JSR oswrch
520 .room RTS
530
540 .sub SEC
550 LDA &06:SBC &70:STA &70
560 LDA &07:SBC &71:STA &71
570 RTS
580 .inc CLC
590 LDA &02:ADC #lobyte:STA &70
600 LDA &03:ADC #hibyte:STA &71
610 RTS
620
630 ]:NEXT
640 ?&220=start MOD 256
650 ?&221=start DIV 256
660 *FX14,2
670 ENDPROC
```

bit calculation, subtracting the contents of &70 and &71 from HIMEM. The value of HIMEM is stored on zero page in locations &06 and &07. Note here how absolute memory locations have been used — mainly because these locations have had these uses in all of the versions of BBC BASIC.

After this subtraction has been performed, line 490 checks the sign of the result. If it is a positive result, it indicates that there is still more than the previously set threshold of bytes of memory remaining. If the result is negative, there is insufficient memory left, and so it's time to make a noise about it. The OSWRCH with A=7 call is then made to cause the beep.

The two programs we have discussed so far are simple examples of potentially useful applications. One advantage of using the operating system calls directly is that we can generate event-driven code — a facility unavailable in BASIC. The following programs both use events.

Background Music

```
 10 REM **** SOUND EVENTS ****
 50
 60 PROCassemble
 80 CALL initialise
 90 END
100
110 DEFPROCassemble
120 note_count=&70
130 oswrch=&FFEE
140 osbyte=&FFF4
150 osword=&FFF1
160
170 FOR pass=0 TO 2 STEP 2
180 P%=&0C00
190
200 [OPT pass
210 .initialise
220 LDA #event MOD 256:STA &220
230 LDA #event DIV 256:STA &221
240 LDA #14:LDX #5:JSR osbyte
250 LDA #0:STA note_count
260 JSR clock
270 RTS
280
290 .event
300 PHP
310 PHA
320 TXA:PHA
330 TYA:PHA
340
350 JSR play_note
360 JSR clock
370
380
390 PLA:TAY
400 PLA:TAX
410 PLA
420 PLP
430 RTS
440
450 .play_note
460 LDY note_count
470 .sloop LDA notetable,Y
480 STA soundtable+4
490 JSR sound
500 INC note_count
510 LDA note_count:CMP#5:BNE out
520 LDA #0:STA note_count
530 .out RTS
540
550
560 .clock
570 LDX #time MOD 256
580 LDY #time DIV 256
590 LDA #4
600 JSR osword
610 RTS
620
630 .sound PHA
640 TXA:PHA
650 TYA:PHA
660 LDX #soundtable MOD 256
670 LDY #soundtable DIV 256
680 LDA #7
690 JSR osword
700 PLA:TAY
710 PLA:TAX
720 PLA
730 RTS
740
750 .notetable
760 EQUD &00000000
770 EQUD &00000000
780 .soundtable
790 EQUD &00000000
800 EQUD &00000000
810 .time EQUD &FFFFFFCF
820 EQUB &FF
830 ]:NEXT
840 FOR I%=0 TO 4:READ data:?
    (notetable+I%)=data:NEXT
850 FOR I%=0 TO 7:READ data:?
    (soundtable+I%)=data:NEXT
860 ENDPROC
870 DATA 69,73,81,89,97
880 DATA 1,0,&FA,&FF,0,0,10,0
```

BACKGROUND MUSIC

This program uses the Interval timer has 'timed out' event to play a series of musical notes while the computer performs other tasks. As the event routine runs in the background, you can save a program, edit it, list it or run a different program, while the notes are played throughout. You could expand this program to play a tune while a game program runs, for example. The tune will even be played during disk operations!

The program is assembled to start at address &0C00. The first section is an initialisation routine that performs the following tasks. The event vector EVENTV, at addresses &220 and &221, is changed to point to our routine. OSBYTE 14 with X=5 is then called to enable the Interval timer has 'timed out' event. A 'note counter' at location &70 is initialised to zero, and finally the event timer is started by a call to the clock subroutine.

The first task of the event-handling routine is to save the CPU registers on the stack. A subroutine is then called to play a single note and the timer clock is reset. Finally, the registers are restored. One thing to note here is that the register save and restore parts of the program are standard to all our event-handling routines, and we simply change the code between these two blocks of instructions.

Let's now examine the various parts of the subroutine that play the notes. One note is played, selected from a table of notes, every time the event timer crosses zero. The table of notes, called notetable, is simply a series of bytes holding the pitch information for the five notes that we want to play. A counter is used to determine which of the five notes is to be played, and the value is held in &70. The pitch value is retrieved and then stored at the appropriate place within soundtable, which is simply the parameter block for the OSWORD routine that we will later use to generate the sound. The counter is then incremented, thus pointing it to the next position within the note table for the next time the routine is entered. As you've only got five notes in the table, as soon as location &70 holds the value five it is reset to zero.

The subroutine clock uses OSWORD with A=4 to set the event timer running. The area of memory called time holds the value that is to be stored in the timer registers, thus specifying the time between the notes played. The sound subroutine performs the equivalent of a SOUND command, using OSWORD with A=7; soundtable is the parameter block for this OSWORD call. The data for notetable and soundtable is set up by the BASIC statements in lines 840 and 850.

Pressing BREAK will stop this program, by resetting the EVENTV contents. However, calling the initialise routine will set things off again.

GRAPHICS MOVEMENT

Our final program shows how we can use events to move a simple graphics shape, in this case a rectangle, across the screen, one position for each event. The frequency of movement therefore depends upon the value put into the event timer. This value is set up in line 1260. The definition of the shape to be moved, in terms of PLOT numbers, is stored in shapetable and is set up from a DATA statement — line 1220 POKEing the appropriate values in. Lines 1230 to 1250 set up EVENTV to point to our program and enable the appropriate event. Now let's consider the machine code involved.

This is stored in memory at the address specified by code%. Lines 200 to 230 do the usual register save operation, and lines 250 to 280 do the work by a series of subroutine calls. Finally, we restore the registers and execute an RTS. Lines 370 to 440 save the current status of the graphics cursor and the current GCOL in use so that it can be restored after we've handled the event.

Lines 460 to 520 deal with the movement of the shape. Lines 540 to 630 restore the graphics

Graphics Movement

```
 10 REM **** GRAPHICS USING  EVENT TIMER ****
 30 MODE 1
 40 PROCassemble
 50 CALL clock
 60 END
 70 DEFPROCassemble
 80 xpos=&70:ypos=&72
 90 ?xpos=100:?(xpos+1)=0
100 ?ypos=100:?(ypos+1)=0
110 oswrch=&FFEE
120 osbyte=&FFF4
130 osword=&FFF1
140 DIM code% 600,time% 10
150
160 FOR pass=0 TO 2 STEP 2
170 P%=code%
180
190 [OPT pass
200 PHP
210 PHA
220 TXA:PHA
230 TYA:PHA
240
250 JSR preserve
260 JSR draw
270 JSR restore
280 JSR clock
290
300
310 PLA:TAY
320 PLA:TAX
330 PLA
340 PLP
350 RTS
360
370 .preserve
380 LDX #pgpos MOD 256
390 LDY # pgpos DIV 256
400 LDA #&0D
410 JSR osword
420 LDA &35B:STA ccol
430 LDA &35C:STA ccol+1
440 RTS
450
460 .draw
470 JSR gcol
480 JSR move
490 JSR shape
500 JSR incx
510 JSR incy
520 RTS
530
540 .restore
550 LDA ccol:STA &35B
560 LDA ccol+1:STA &35C
570 LDA #25:JSR oswrch
580 LDA #4:JSR oswrch
590 LDA cgpos:JSR oswrch
600 LDA cgpos+1:JSR oswrch
610 LDA cgpos+2:JSR oswrch
620 LDA cgpos+3:JSR oswrch
630 RTS
640
650 .move
660 LDA #25:JSR oswrch
670 LDA #4:JSR oswrch
680 LDA xpos:JSR oswrch
690 LDA xpos+1:JSR oswrch
700 LDA xpos+2:JSR oswrch
710 LDA xpos+3:JSR oswrch
720 RTS
730
740 .shape
750 LDX#24:LDY #0
760 .sloop LDA shapetable,Y
770 JSR oswrch
780 INY:DEX
790 BNE sloop
800 RTS
810
820 .incx CLC
830 LDA #2:ADC &70:STA &70
840 LDA #0:ADC &71:STA &71
850 RTS
860 .incy CLC
870 LDA #2:ADC &72:STA &72
880 LDA #0:ADC &73:STA &73
890 LDA &73:CMP #3:BCS overflow
900 RTS
910 .overflow
920 LDA #0:STA &72:STA &73
930 LDA #0:STA &70:STA &71
940 RTS
950
960 .clock
970 LDX #time% MOD 256
980 LDY #time% DIV 256
990 LDA #4
1000 JSR osword
1010 RTS
1020
1030
1040 .gcol
1050 LDA #18:JSR oswrch
1060 LDA #3:JSR oswrch
1070 LDA #1:JSR oswrch
1080 RTS
1090
1100
1110 .ccol EQUD &0000
1120 .pgpos EQUD &00000000
1130 .cgpos EQUD &00000000
1140 .shapetable
1150 EQUD &00000000
1160 EQUD &00000000
1170 EQUD &00000000
1180 EQUD &00000000
1190 EQUD &00000000
1200 EQUD &00000000
1210 ]:NEXT
1220 FOR I%=0 TO 23:READ data:?(shapetable+I%)=data:NEXT
1230 ?&220=code% MOD 256
1240 ?&221=code% DIV 256
1250 *FX14,5
1260 !time%=&FFFFFFEF:time%?4=&FF
1270 ENDPROC
1280 DATA 25,1,60,0,0,0,25,1,0,0,60,0,25,1,&C4,255,0,0,
     25,1,0,0,&C4,255
```

colours and the graphics cursor position to the state they were in when we entered the routine.

Lines 650 to 720 use the OSWRCH routine to carry out the equivalent of a BASIC MOVE command to the desired x and y positions. These positions are stored in xpos and xpos+1 for the x co-ordinate and ypos and ypos+1 for the desired y co-ordinate. Lines 740 to 800 draw the shape, again using OSWRCH. The Y register is used as an index register in order to access the table of bytes that represents the shape (shapetable). The bytes are sent as a stream through OSWRCH.

As the shape moves across the screen, it's clear that there must be routines to update the x and y positions each time the event occurs. This is done by the routines incx and incy in lines 820 to 900. Once the box reaches the top of the screen, the routine overflow resets the x and y co-ordinates to zero. The routine clock uses OSWORD with A=4 to set the event timer off, and the routine gcol does the equivalent operation to a BASIC GCOL3,1

There are several problems associated with this program. Run the code, and as long as the screen doesn't scroll, the shape will move rather uncertainly across the screen. You can perform other operations while this is happening, and all is well. However, speed up the movement, by increasing the value in the time area of memory, and although the shape will move when you list part of the program, there could well be unusual characters appearing on the screen. The reason for this is simple — conflict between our routine, which is entered when an event occurs, and the usual OS behaviour when listing. Both require the use of OSWRCH and what happens is that when the OS listing routine is interrupted by our event, OSWRCH is left in a state that the listing routine is not expecting.

If OSWRCH is interrupted, and is called by the interrupting routine, then one exit from the interrupting routine's OSWRCH call is unavailable. OSWRCH is said to be *non-reentrant.* This is why Acorn suggest that it is unwise to use OS routines in event- and interrupt-handling routines. You *can* use them, but you are cautioned to proceed with care. These problems could be overcome by not using any of the OS routines while in an interrupt- or event-handling routine. There are alternatives, such as entering data directly into the screen RAM area to draw graphics, and this is one of the few times where side-stepping ROM routine calls is a good idea.

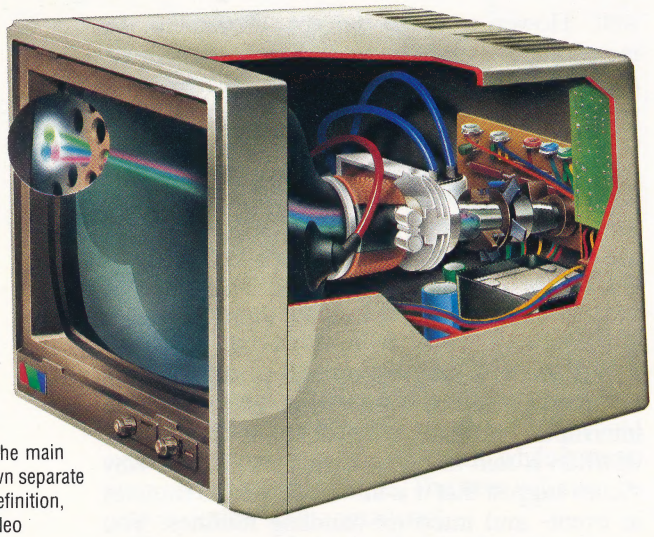
M

MONITOR

In hardware terms, a *monitor* is a visual display unit similar to an ordinary cathode ray tube television except that it has no channel selecting facility — it is a dedicated machine to provide a visual display of the video signal from a computer. Monitors produce a much better picture than a standard television screen because the signals from the computer do not have to be 'modulated', or adapted to mimic the signal that the television receives from an aerial. Instead, a monitor can process the signal directly into the picture that appears on the screen.

The two most popular types of signals used in monitors are 'composite video' and 'RGB'. A composite video monitor generally has a bayonet-type socket fitted that is a common feature on many video recorders. The colour signals arrive from the computer together and the colours are decoded within the monitor itself. An RGB (red, green, blue) monitor signal is usually sent down a DIN-type socket although there are variations such as the use of Peri plugs. The signal is decoded into the separate colours before it is sent to the monitor, which are then transmitted along three individual lines. An RGB monitor generally provides a better picture than a composite video monitor.

In software terms, a monitor is a program controlling several independent programs that may be running at the same time. The monitor will allocate priorities in processor and peripheral time and oversee the progress of each of the programs, passing data between them where necessary. Where a monitor is governing an entire system, the term can be referred to as an 'operating system' or 'supervisor'.



Monitoring The Situation
There are two types of monitor: the 'green screen' monochrome monitor (most commonly used on business machines because it is the easiest on the eye, and therefore the most suitable for long periods of accounting or word processing). and the colour monitor. The latter category can be further subdivided into two groups. RGB monitors have each of the main colour signals sent down separate cables for maximum definition, whereas composite video monitors send all the signals down a single line

MOTHER BOARD

The *mother board* is the main printed circuit board within a computer. It contains all of the major components and systems that are used and often functions without the need for any other boards. The components may consist of the central processing unit, the main memory and the input/output system. Mother boards generally contain slots which allow other boards to be inserted. These additional boards, known as 'daughter boards', contain upgrade modules that perform specific additional functions, such as providing additional RAM, an 80-column or colour card, or a printer or disk interface.

In the early days of home computing, when computers were the preserve of electronics enthusiasts, fitting additional boards to a computer became something of a craze as hobbyists competed with each other to customise their machines. It reached the stage where owners of Apple computers could no longer fit the lids on their computers because of all the boards that had been added! Although this still occurs among many enthusiasts, the main customers for daughter boards today are businesses who require additional boards to run specific applications on their personal computers.

MOUSE

Over the past few years the term *mouse* has come to mean two completely different things. To the home micro user, it refers to a device that is moved around a desk top, which in turn moves a corresponding cursor around the computer's screen. When the cursor is above a command or an icon, the user presses a button on the mouse. This sends a RETURN signal to the computer, which then executes the command indicated.

Typically, this device works by having a ball bearing on the bottom of the mouse and a shaft encoder attached to a pair of encoding wheels in contact with the ball bearing. As the bearing rolls across the desk, its movements will be mirrored by the wheels and transmitted through the shaft encoder to an electrical sensor, which sends a digital signal to the computer.

Until 1984, a mouse was considered something of a gimmick among computer users. However, with the launch of the Apple Macintosh, the industry began to realise how useful these devices could be and by 1985 there was a rush to produce Macintosh mouse-type packages for an increasing number of machines (see page 1050).

A mouse can also refer to an 'intelligent' robotic device that can find its way around a previously unseen maze. Generally, the mouse will have some kind of sensor system on the front to detect collisions, and will either have a system of on-board logic, enabling it to decide in which direction to move next, or else it will be attached to a computer that will provide the processing for it.

The mouse moves through a maze until it encounters a wall, at which point it will attempt to move in a different direction until it finds a way through to the centre of the maze. On its first run, the mouse will 'remember' the directions it has travelled, and should be able to find the quickest route through the maze on the second. Robot mice are becoming increasingly popular with robotics enthusiasts and there are now world-wide championships, held annually, to find the world's fastest mouse (see page 721).

HAVE YOU ORDERED YOUR VOLUME FIVE BINDER YET?

FOLD 2

IF YOU HAVE NOT ASKED FOR THE BINDERS TO BE SENT TO YOU AUTOMATICALLY, DO SO NOW, AND ENSURE THAT YOUR COPIES OF THE HOME COMPUTER ADVANCED COURSE ARE KEPT PROPERLY BOUND AND IN GOOD CONDITION FOR YEARS TO COME.

BY TICKING THE BOX OPPOSITE, YOU WILL BE SENT BINDER NUMBER 5.

☐ PLEASE SEND ME MY VOLUME 5 BINDER NOW. I ENCLOSE A CHEQUE/POSTAL ORDER FOR £3.95 (WHICH INCLUDES POSTAGE AND PACKING).

FOLD 3

BY TICKING THE OTHER BOX *AS WELL*, YOU WILL BE SENT SUBSEQUENT BINDERS FOR YOUR COLLECTION.

☐ I WOULD ALSO LIKE TO RECEIVE FUTURE BINDERS AS THEY ARE ISSUED. I UNDERSTAND THAT I WILL RECEIVE A PAYMENT ADVICE FOR £3.95 (WHICH INCLUDES POSTAGE AND PACKING) WITH EACH BINDER. IF I AM NOT SATISFIED WITH THE BINDER, I CAN RETURN IT TO YOU, WITHIN 14 DAYS, AND OWE NOTHING.

NO STAMP NECESSARY.

JUST FOLD UP THE PAGE AS INDICATED, REMEMBERING TO ENCLOSE YOUR CHEQUE/POSTAL ORDER MADE PAYABLE TO ORBIS PUBLISHING, AND SEND TO US TODAY.

FOLD 4

FOLD 4

I enclose a cheque/postal order made payable to: Orbis Publishing Ltd. for a total of £____ which I understand includes the cost of postage and packing.

NB: Please allow 28 days for the delivery of your binders.

When you have completed the order form fill in your name and address in the space provided.

Then cut along the dotted line to detach the page, enclose your cheque/postal order, and fold the page carefully – following the instructions to complete the reply paid envelope.

NO STAMP NECESSARY.

IF YOU ALREADY HAVE AN ACCOUNT NO. FOR YOUR BINDERS PLEASE FILL IT IN HERE.

Complete the section below with one letter or figure per space.

MR		INITIALS	SURNAME
MRS			
MISS			

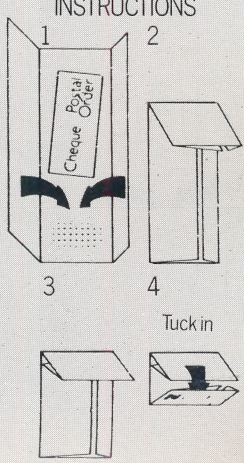
ADDRESS & POST CODE

TELEPHONE NO. INCLUDING STD CODE OR EXCHANGE NAME

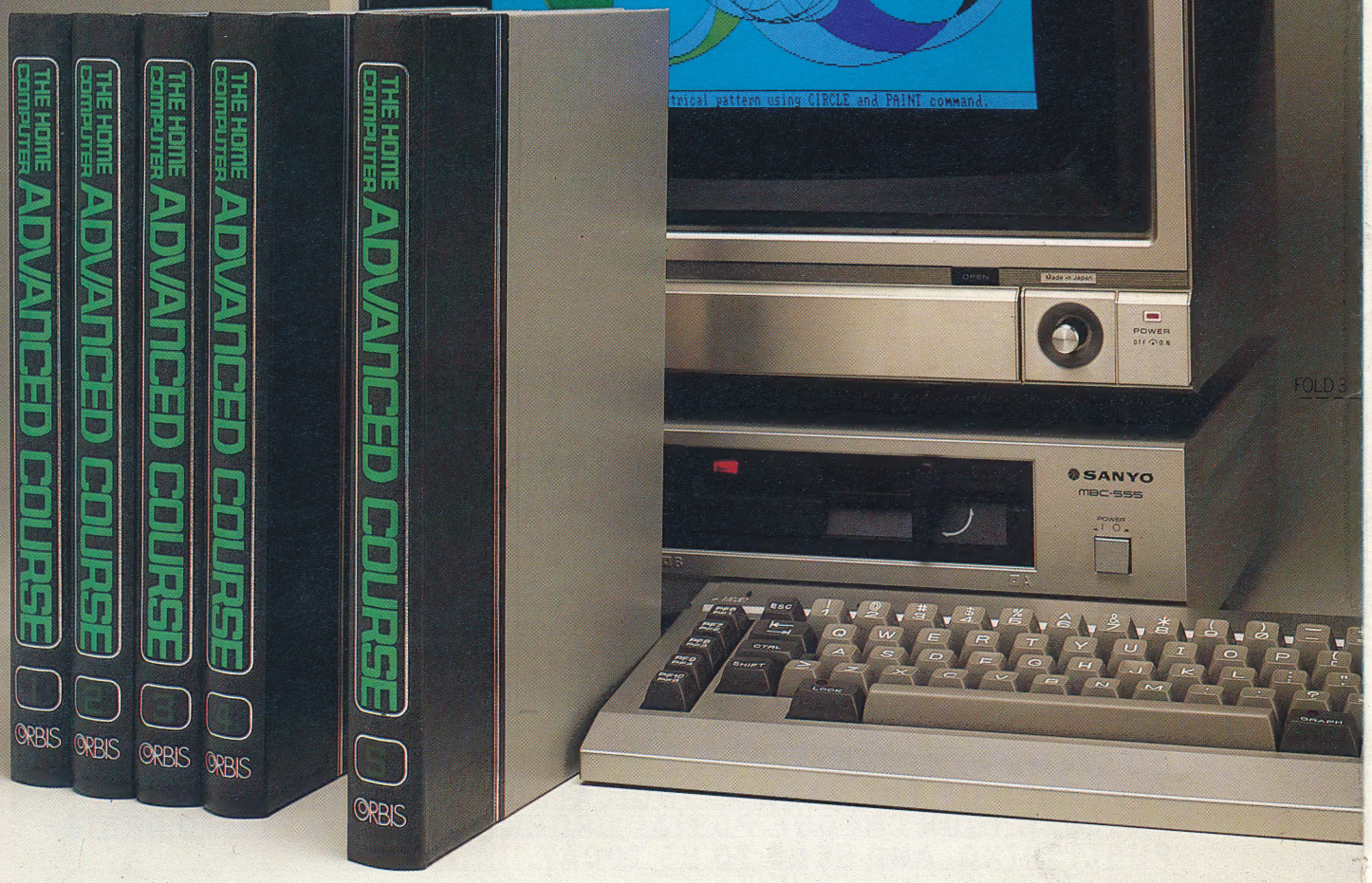
FOLD 1

FOLD 1

FOLD 4
FOLDING INSTRUCTIONS



ARE YOU LOOKING AFTER YOUR COPIES.

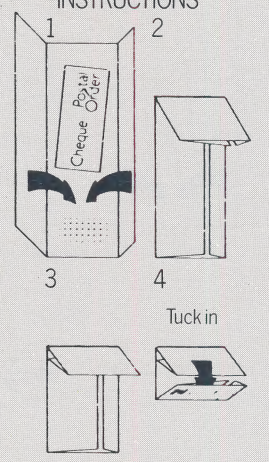


FOLD 2

FOLD 3

FOLD 4

FOLD 4
FOLDING INSTRUCTIONS



FOLD 1

Postage will be paid by licensee

Do not affix Postage Stamps if posted in Gt. Britain, Channel Islands or N. Ireland.

BUSINESS REPLY SERVICE
Licence No. SW4035.

The Home Computer
Advanced Course Binders
Orbis House, 20-22 Bedfordbury
London WC2N 4BR

|2|

FOLD 1

GUARANTEE GUARANTEE GUARANTEE GUARANTEE GUARANTEE

GUARANTEE

If you are not entirely satisfied with your binder, send it back immediately and it will be either exchanged, or, if you prefer, your money will be refunded in full.

GUARANTEE

GUARANTEE GUARANTEE GUARANTEE GUARANTEE GUARANTEE